Titre: Title:	Towards Station-Level Demand Prediction for Effective Rebalancing in Bike-Sharing Systems
Auteur: Author:	Pierre Hulot
Date:	2018
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Hulot, P. (2018). Towards Station-Level Demand Prediction for Effective Rebalancing in Bike-Sharing Systems [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/3160/

UNIVERSITÉ DE MONTRÉAL

TOWARDS STATION-LEVEL DEMAND PREDICTION FOR EFFECTIVE REBALANCING IN BIKE-SHARING SYSTEMS

PIERRE HULOT
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DILPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
MAI 2018

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

TOWARDS STATION-LEVEL DEMAND PREDICTION FOR EFFECTIVE REBALANCING IN BIKE-SHARING SYSTEMS

présenté par : HULOT Pierre
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. BILODEAU Guillaume-Alexandre, Ph. D., président
M. ALOISE Daniel, Ph. D., membre et directeur de recherche
M. JENA Sanjay Dominik, Ph. D., membre et codirecteur de recherche
M. LODI Andrea, Ph. D., membre et codirecteur de recherche
M. ADULYASAK Yossiri, Ph. D., membre externe

ACKNOWLEDGEMENTS

I would especially like to thank my research director Professor Daniel Aloise and my co-director Sanjay Dominik Jena for guiding me and supporting me throughout my master's degree, and for helping me to target my work.

I also thank Bixi, by the people of Nicolas Blain and Antoine Giraud, for quickly providing me the data and for following the project with interest.

I would also like to thank the chair of "data science for real-time decision making", and especially Andrea Lodi, who funded this project. I also thank the partners of the Chair : GERAD, IVADO and CIRRELT.

I would also like to thank the postdocs, PhD students and masters of the chair who have been assisting me throughout this work and helping me solve my problems.

Finally, I thank the Department of Génie Informatique et Génie Logiciel (GIGL) of Polytechnique Montreal

RÉSUMÉ

Les systèmes de vélos en libre-service sont utilisés à l'échelle mondiale pour soulager la congestion et apporter une solution aux problèmes environnementaux dans les villes. De nos jours, presque toutes les grandes villes ont un tel système. Ces systèmes sont très pratiques pour les utilisateurs qui n'ont pas besoin de faire l'entretien du vélo et peuvent le rendre presque partout dans la ville. Cependant, le nombre croissant d'abonnés et la demande aléatoire rendent la planification opérationnelle du système très difficile. Prédire la demande en vélos a été l'objet de nombreuses recherches dans la communauté scientifique. Cependant, la plupart des travaux ont cherché à prédire la demande globale du réseau, qui n'est généralement pas suffisante pour améliorer la planification opérationnelle. En effet elle nécessite des prévisions spécifiques pour chaque station et à des moments précis de la journée. Ces travaux ont montré qu'une variation significative du trafic peut être liée à des comportements réguliers, et à des facteurs externes tels que les heures de pointe ou les conditions météorologiques. En particulier, de nombreux opérateurs utilisent des intervalles pour combler les lacunes dans la prédiction du trafic. Cependant, très peu de travaux ont cherché à correctement définir ces intervalles.

Dans cette recherche, nous nous concentrons sur la modélisation de la distribution statistique du nombre de déplacements qui se produisent à chaque heure et chaque station. Ce modèle ne se contente pas de prédire l'espérance du nombre de voyages prévus, mais aussi la probabilité de chaque nombre de départs et d'arrivées par station en utilisant la demande historique. Le modèle mis en place est composé de trois parties. Tout d'abord, nous estimons, en utilisant des techniques d'apprentissage machine, le nombre de trajets attendus à chaque station. Puis, nous calculons la confiance sur la première prédiction (variance attendue). Enfin, nous déterminons la bonne distribution à utiliser.

La première partie (espérance du nombre de voyages) utilise un algorithme en deux étapes qui d'abord réduit le problème à un problème plus simple, minimisant la perte d'information, puis un algorithme prédictif est appris sur le problème réduit. Le processus de prédiction de la demande (utilisation du modèle) inverse ce mécanisme. Plusieurs méthodes de simplification et de prédiction sont testées et comparées en termes de précision et de temps de calcul (RMSE, MAE, erreur proportionelle, R^2, vraisemblance). Les résultats montrent que le choix du meilleur algorithme dépend de la station. Un modèle combinant les modèles précédents est proposé. Ces algorithmes utilisent la demande des stations voisines pour améliorer leurs performances en les agrégeant. Ils extraient des stations des comportements principaux qui sont

ensuite modélisés. Ces comportements sont alors plus simples et stables (moins aléatoires).

Les conclusions de ce travail sont validées sur les réseaux de Montréal, New York et Washington. Enfin, ce modèle est utilisé pour définir une stratégie de rééquilibrage en temps réel pour Bixi à Montréal, en raffinant la définition des intervalles de rebalancement.

ABSTRACT

Bikesharing systems are globally used and provide relief to congestion and environmental issues in cities. Nowadays, almost all big cities have a bicycle-sharing system. These systems are very convenient for users that don't need to do maintenance of the bicycle and can return it almost everywhere in the city. However, the increasing number of subscribers and the stochastic demand makes the operational planning of the system very difficult. Predicting bike demand has been a major effort in the scientific community. However, most of the efforts have been focused on the prediction of the global demand for the entire system. This is typically not sufficient to improve the operational planning, which requires demand predictions for each station and at specific moments during the day. A significant variation of the traffic can be linked to regular behaviors, and external factors as peak hours or weather. In particular, many system operators use fill level intervals which guide the redeployment crews in their efforts to equilibrate the system. However, little work has been done on how to effectively define those fill levels.

In this research, we focus on modeling the distribution of the number of trips that occur at each hour and each station. This model not only seeks to predict the number of expected trips, but also determines as precisely as possible the expected distribution of trips. It uses the historical observed demand to predict future demand. The prediction model is composed of three parts. First, we estimate from historical data the expected number of trips, using machine learning techniques that use features related to weather and time. Second, we compute the confidence of the first prediction (expected variance). Finally, we focus on determining the right distribution to use. The first part uses a two-step algorithm that first reduces the problem to a simpler one, minimizing the information lost, then learns a predictive algorithm on the reduced problem. The prediction process inverts this mechanism. Several simplification and prediction methods are tested and compared in terms of precision and computing times. The final test compares distribution estimations in terms of log likelihood. The results show that the choice of the best algorithm depends on the station. Then a combined model is proposed to better model the demand. Our models are tested on several networks (Montreal, New York and Washington). Finally, this model is used to define an online rebalancing strategy close to the one used by Bixi at Montreal. This strategy has been deployed in Montreal.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii

RÉSUMÉ iv

ABSTRACT vi

TABLE OF CONTENTS vii

LIST OF TABLES x

LIST OF FIGURES xi

LIST OF ACRONYMS AND ABBREVIATIONS xiii

LIST OF APPENDICES xiv

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 LITERATURE REVIEW 5
2.1 Classification of problems in bike-sharing systems 5
2.2 First Level : Network Design 5
2.3 Second level : Traffic analysis 7
2.4 Second Level : Estimating Lost Demand 10
2.5 Third level : Redistribution 10
2.6 Third Level : Targets for Rebalancing 12
2.7 Summary 12

CHAPTER 3 DATA ANALYSIS 16
3.1 Bixi's Data 16
3.1.1 Bixi private data 18
3.2 Weather data 19
3.3 Holidays and National Days 20
3.4 Preprocessing 20
3.4.1 Features preprocessing 20
3.4.2 Objective preprocessing 21
3.5 Missing Data 22

3.6 Feature Analysis . . . 23
3.6.1 Feature significance . . . 28
3.6.2 Data analysis conclusions . . . 29

CHAPTER 4 MODELING DEMAND PER STATION . . . 30
4.1 General Overview . . . 30
4.2 Evaluating models . . . 32
4.2.1 Splitting Data . . . 33
4.2.2 Size of Stations . . . 33
4.2.3 MAE . . . 33
4.2.4 RMSE . . . 34
4.2.5 MAPE . . . 34
4.2.6 RMSLE . . . 34
4.2.7 R^2 Score . . . 34
4.3 Predicting Traffic Expectation . . . 36
4.3.1 Literature . . . 38
4.3.2 Reduction Methods . . . 38
4.3.3 Reconstruction loss and reduction optimization . . . 43
4.3.4 Prediction Methods . . . 48
4.4 Expectation Hyperparameters Optimization . . . 51
4.4.1 Selection of the number of dimension, optimization of reduction methods 51
4.4.2 Linear Regression . . . 52
4.4.3 Decision Tree . . . 53
4.4.4 Random Forest . . . 53
4.4.5 Gradient Boosted Tree . . . 54
4.4.6 Multi-Layer Perceptron (MLP) . . . 54
4.5 Test of mean traffic estimation model . . . 55
4.5.1 Test of Data Transformations . . . 55
4.5.2 Features importance analysis . . . 62
4.5.3 Analysis of the results of the proposed algorithms . . . 64
4.5.4 Ensemble Model for mean estimation . . . 65
4.6 Predicting Traffic Variance . . . 69
4.6.1 Variance hyperparameters optimization . . . 71
4.6.2 Variance estimation results . . . 73
4.7 Fitting the Trip Distribution . . . 74
4.7.1 Comparing distributions : the Log Likelihood . . . 74

4.7.2 Poisson Distribution . . . 75
4.7.3 Negative Binomial Distribution . . . 76
4.7.4 Zero Inflated Poisson . . . 76
4.7.5 Distribution Results . . . 77
4.8 Results on the Test dataset . . . 80
4.8.1 General Results . . . 80
4.8.2 Results per stations : stations 6507, 5005, 6221 . . . 84
4.8.3 Geographical distribution . . . 87
4.8.4 Adding new stations . . . 89
4.9 Application on other bike-sharing networks . . . 89
4.9.1 Montreal . . . 89
4.9.2 Washington . . . 90
4.9.3 New York . . . 91
4.9.4 Conclusions . . . 91

CHAPTER 5 GENERATING DECISION INTERVALS FOR REBALANCING . . . 93
5.1 Service Level . . . 93
5.1.1 Modification of the service level . . . 94
5.1.2 Time Horizon . . . 95
5.1.3 Implementation Notes . . . 95
5.1.4 Analysis of the Service Level . . . 95
5.2 Defining decision intervals . . . 96
5.2.1 Test of decision intervals . . . 98
5.2.2 Results on the decision intervals . . . 100
5.2.3 Estimating the marginal improvement of increasing the rebalancing capacity . . . 103

CHAPTER 6 GENERAL DISCUSSION . . . 106

CHAPTER 7 CONCLUSION . . . 110

REFERENCES . . . 113

APPENDICES . . . 119

LIST OF TABLES

Table 2.1	Contribution in bike-sharing system rebalancing	13
Table 2.2	Main contribution is traffic modeling	14
Table 3.1	Missing data numbers	22
Table 3.2	gap occurrences	23
Table 3.3	Available features for the model	23
Table 3.4	Selected features for the model	29
Table 4.1	Summary of principal characteristics of reduction methods	47
Table 4.2	correlations between $RMSE$ score of four different algorithms, varying the random forest hyperparameters.	51
Table 4.3	Precision of the log transformation compared to the normal data	56
Table 4.4	Impact of the decorrelation on the precision	56
Table 4.5	Impact of the normalization on the precision	57
Table 4.6	average performance of time augmented models	60
Table 4.7	Precision and running time of mean estimators	64
Table 4.8	Comparison of reduction methods, using a linear predictor	65
Table 4.9	Results of combined algorithms	68
Table 4.10	Performance of estimators on variance after optimization	72
Table 4.11	Precision of variance estimators (validation set)	73
Table 4.12	Average Log likelihood per distribution hypothesis of several models	78
Table 4.13	Proportion of votes per hypothesis	78
Table 4.14	scores of each model on the test set	83
Table 4.15	Scores on three stations	85
Table 4.16	Scores on Montreal (Bixi) without weather features	90
Table 4.17	Scores on Washington (Capital Bikeshare)	90
Table 4.18	Scores on CityBike (New York)	91
Table 5.1	average sum of minimums, targets and maximums per hour	99
Table 5.2	Decision interval scores	102
Table 5.3	Influence of α value	104
Table A.1	Precision scores for mean estimation algorithms on validation set	119
Table B.1	Scores on Montreal (Bixi) without weather features	123
Table B.2	Scores on Washington (Capital Bikeshare)	124
Table B.3	Scores on CityBike (New York)	125

LIST OF FIGURES

Figure 3.1 Example of a trip file 17
Figure 3.2 Example of precipitation file 19
Figure 3.3 Repartition of traffic during the week (Monday to Sunday) 21
Figure 3.4 Trip Numbers per Feature 26
Figure 3.5 Correlation matrix 27
Figure 3.6 ANOVA results 28
Figure 4.1 Model preview 31
Figure 4.2 Model for mean prediction 36
Figure 4.3 Explained variance ratio per singular value 44
Figure 4.4 SVD first components 45
Figure 4.5 Kmeans analysis 46
Figure 4.6 $RMSE$ per dimension for kmeans and SVD 52
Figure 4.7 Scores evolution per time window 58
Figure 4.8 First time dependent model 59
Figure 4.9 Second time dependent model 59
Figure 4.10 R2 score during the jazz week (first figure) and a September week (second figure), for hours between 2h p.m. and 2h a.m. 61
Figure 4.11 Feature importance for decision tree algorithms 63
Figure 4.12 Number of stations for which each algorithms is the best 66
Figure 4.13 Medium size of stations per best algorithm 67
Figure 4.14 Votes for algorithms (orange first vote, blue second vote) 68
Figure 4.15 Residuals versus features 70
Figure 4.16 Votes for best algorithms, based on the best $RMSE$ 79
Figure 4.17 Error Deviation 82
Figure 4.18 Station size versus MAPE 84
Figure 4.19 Station size versus R2 85
Figure 4.20 Predicted and real Traffic in station 6221 86
Figure 4.21 Predicted and real Traffic in station 6307 87
Figure 4.22 Predicted and real Traffic in station 5005 88
Figure 4.23 Geographical distribution of the error 88
Figure 4.24 Station sizes per network 92
Figure 5.1 Evolution of the service level with different time horizons, from 2 hours (gray) to 100 hours (black) 96

Figure 5.2 Selection of station's service level in the same period 96
Figure 5.3 Rules to compute decision intervals on station 46 and 202 98
Figure 5.4 Evolution of scores in respect to β ($\alpha = 0.5$) 102
Figure 5.5 Evolution of scores in respect to α ($\beta = 0.65$) 103
Figure 5.6 Marginal utility of rebalancing 104
Figure 5.7 Distribution of rebalancing operation during the day 105
Figure 7.1 Adaptive model flowsheet, on the top left box the learning of the predictive algorithm, on the top right box the learning of the adaptive part and the bottom the global prediction process 111

LIST OF ACRONYMS AND ABBREVIATIONS

RL	Reconstruction Loss
RL_{norm}	Normalized Reconstruction Loss
$MAPE$	Mean Absolute Percentage Error
$RMSE$	Root Mean Squared Error
$RMSLE$	Root Mean Squared Logarithmic Error
MAE	Mean Absolute Error
R^2	Coefficient of determination
ANOVA	ANalysis Of Variance
arr	arrival
dep	departure
SVD	Singular value decomposition
GM	Gaussian Mixture
id	Identity
GBT	Gradient Boosted Tree
DT	Decision Tree
AE	AutoEncoder
RF	Random Forest
c_n	Ensemble model with the best n algorithms

LIST OF APPENDICES

Annexe A PRECISION RESULTS 119
Annexe B RESULTS ON MONTREAL, WASHINGTON AND NEW YORK . . 123

CHAPTER 1 INTRODUCTION

Bikesharing systems are widely used and offer an appropriate solution to congestion in big cities. Today almost all big cities have their own bikesharing system. Three generations of systems led to an efficient and reliable one, which does not suffer from thefts and deterioration. These systems are very convenient for users that don't need to do the maintenance of the bicycle and can return it almost everywhere in the city. However, the increasing number of subscribers and variability of the demand makes the operation and maintenance of the system very hard.

In this context, traffic prediction and network rebalancing are the main operational goals of the system operator. The operator's role is to ensure that it is always possible to rent or return a bicycle in every station. However, people behavior is variable and hard to predict. The operator also needs to consider the natural trend of the network to unbalance. These two difficulties are worsened by the difference in response times. While it takes less than one hour for the network to unbalance, the operator needs more than half a day to visit all stations once. Thus, the operator needs to anticipate three to six hours before the evolution of the system, to be able to prepare it. Therefore the operator is interested into modeling the traffic using exogenous factors as the time and the weather.

Some work has been done to optimize rebalancing strategies but failed to provide a solution applicable by the industry. Besides, today traffic prediction is mostly done using empirical prediction and operators experience. Thus, rebalancing is performed using greedy strategies based on the empirical targets. Traffic has been analyzed from a global point of view, but the literature on traffic prediction per station is rare.

This work explores several traffic models and selects the more precise and efficient one. It studies the dependencies between the bike traffic and some exogenous features as time and weather. This work also proposes an online rebalancing strategy that uses the traffic model to improve the service level of the operator. This strategy is inspired from the operator's one and is directly applicable to real world problems.

Bixi Montreal

This work has been done in partnership with Bixi Montreal, the operator of the Montreal system. This section presents the environment and challenges of this project. Bixi is a network of 5220 bicycles and 462 stations (2016). It has been created by order of the city of Montreal,

and has been inspired by similar models, in European cities. The initiative was part of the transportation plan "reinventer Montreal" of 2008. PBSC took the responsibility of developing and deploying the network. They used the technology already used for parking slots. Bixi is a relatively big network. It began in 2009 with 3000 bicycles and 300 stations, deployed mainly in downtown and in the densely populated areas. 160 stations and 2000 bicycles were added in 2010 to complete the system and to reach the 2016 one. In 2017 Bixi expanded its network to achieve 6000 bicycles and 540 stations. This work is based on the 2016 network.

Bixi is part of the third generation of bikesharing systems. The first generation used free bicycles painted in a specific color that anybody could rent and return, but due to technical limitations (theft and deterioration), these systems were closed or updated. The second-generation systems introduced docks and money deposits to rent bicycles. Bicycles were also designed to resist to vandalism. However, these systems are more expensive than the first one, and are financed by municipalities. Though thefts continued, and the service level dropped, the third generation introduced limited time rent and used wireless technology to follow in real time the evolution of the network, considerably improving the service level.

Bixi has 35 000 members and 235 000 users that make 4 million trips each year, using 5 to 6 times each bicycle a day. Then rebalancing optimization is a priority for the operator. Bixi uses 10 trucks of a capacity of 40 bicycles to counter the unbalance generated by users. They are able to tracks in real time the network status using wireless (solar powered and wireless communication) and movable stations of variable capacity. Montreal traffic is also limited by the weather, during winter it snows about 200 cm in total. The network is then closed between November and April.

Bixi deployment was very successful and their system was exported to other cities as London (687 stations), Minneapolis (146), Washington (252), New York (600) and Chicago (400). The system is also used in Ottawa (25), Melbourne (52), Toronto (80), Boston (90), Chattanooga (30) and Pittsburgh (50).

Bixi challenge today is to assure the best service level for their users, optimizing their re-balancing operations. For that purpose, it uses a handmade decision process for rebalancing based on acceptable fill levels.

Objectives

This work seeks to model as accurately as possible the bicycle traffic in Montreal. It also provides a methodology and conclusions applicable to other cities. The model tries to define the influence of a maximum of features on the bikesharing traffic. These features are mainly time or weather-related. The model is also built to predict the probability distribution of the number of trips per station, and not only its expectation. To accomplish this goal, this work seeks to show the pertinence of a problem reduction approach that is able to take advantage of station similarities, while being specific to each station.

This work also seeks to propose an application of the model in a real situation. This work uses the model to improve the operator rebalancing strategy by optimizing station inventory.

This work is also intended as a preliminary to a reinforcement learning approach of network rebalancing problem.

Work Overview

This dissertation presents all the work done to build the proposed traffic models and their application to real world problems, as well as its deployment into the Bixi system. This work covers the subject in the following way.

The first chapter presents a literature review that covers most articles about traffic analysis in bikesharing systems. A quick overview of the work done in rebalancing is done to justify the need for accurate traffic prediction. Papers on traffic analysis are classified from their objectives in traffic design and traffic prediction. A gap in traffic prediction is shown, and this work proposes a methodology to fill it. The second chapter presents the available data from Bixi, Citybike, Capital Bikeshare and Données Canada. It presents trip data, weather data, critical event data and station data for each bikesharing system. This data is analyzed to extract the relevant information and transformed to suit the problem requirements. All preprocessing steps are described.

The third chapter presents the proposed model for traffic modelization per station and per hour. This chapter is divided into five sections. The first section presents the architecture and the methodology used to build the model : a two-phased algorithm with a simplification part and a prediction part. The second section presents the first step of the methodology : predicting the traffic expectation. A new method is proposed for multiple prediction, using a simplification of the problem extracting main behavior and predict station traffic from these behaviors. The next section analyzes and predicts traffic variance, by performing a new feature

selection. The fourth section uses the first two predictions (expectation and variance) to fit a statistical distribution on the model (on each station). This section completes the model. The last section compares different algorithms built by this methodology and compare them to state-of-the-art algorithms. All algorithms are compared in terms of precision, rapidity and likelihood.

The fourth chapter presents an application of the model to a real-world problem by automating intervals and target generations. This application improves the performance of Bixi's strategy, giving a mathematical background to the decision intervals used to rebalance the system. These intervals are generated using this probabilistic model and service level. The generated intervals are tested on real data using a worst-case approach. They are compared to Bixi ones.

Finally, the dissertation concludes on the best methods to predict bikesharing traffic expectation, and proposes future work based on these models.

CHAPTER 2 LITERATURE REVIEW

2.1 Classification of problems in bike-sharing systems

A lot of problems are related to bike-sharing systems, the main and most addressed one in the literature is the management of the demand. The main issue with the demand is to address it at every time of the day, every day. The system has a natural trend to unbalance itself because most travels are short and one way (Vogel and Mattfeld, 2010). They also study the traveling causes and identify two main sources of unbalance : continuous sources and discrete sources. Examples of continuous sources are the height difference and peak hours. Discrete sources are for example weather, events, underground failures, which are unpredictable or irregular. The system operator goal is to counter this unbalance to meet the demand by means of very limited resources.
The operator has three decision levels : the first, the long-term level, is used for the design of a new network or an expansion or modification of an existing one. On this level, the aim is to place stations and evaluate potential demand analyzing neighboring effects. The second level is a mid-term level in which the operator fixes hourly objectives based on historical behavior of the system. The fixed objectives can be set every month but also in real time. This part uses knowledge about the network to influence operator behavior. The last level is the operational level, the online level. The operator takes real-time decisions on the network to balance it and uses the analysis and results of the second to better serve the demand.
The problems tackled in each level are :

— **First level** corresponds to the network design

— **Second level** rebalancing planning, demand modelization

— **Third level** rebalancing operations, network maintain

This work proposes a traffic prediction tools (second level). However, the built model uses weather transforming the model into a real-time prediction model, getting it closer to a third-level decision.

2.2 First Level : Network Design

The first level of decision is the network design. This level tries to define the area to be covered, the number of locations and size of stations, the number of bicycles, and evaluates the repositioning possibilities. This problem is very close to the demand prediction problem : the main goal is to explain the demand using external factors, as weather, time, but also

environment (connections, or neighboring activity). Traffic modelization and network design share the same goal : they try to explain the demand from exterior factors. However, the analyzed features are different, the network design problem is more interested in how neighboring activity will influence the demand, whereas in traffic modelization these effects are contained into the station behavior and are static. Then neighboring effects are ignored most of the time for traffic prediction. Only models that aim to handle breakdowns and rare effects take these effects into consideration.

A lot of articles tried to explain the demand from neighboring properties. Mahmoud et al. analyze the effect of socio-demographic attributes, land use, built environment and some weather measures. They show an effect on the distance between stations and the number of neighboring intersections. Faghih-Imani and Eluru (2016) also analyze these effects, and prove a dependence with surrounding restaurants, parks, bike lanes and job and population density. Rudloff and Lackner (2014) in a similar analysis shows that the traffic depends on the neighboring station status, and weather conditions (temperature, humidity, rain). Hampshire and Marla (2012) also analyze the relationship between the bike-sharing traffic and other transportation means. They show that this relationship depends on the hour of the day. Gebhart and Noland (2014) analyze the impact of weather conditions on the bike traffic. Vogel and Mattfeld (2011) analyze the relationship between time and spacial imbalancement due to one-way trips and short rental times. Vogel et al. (2011) uses the time features to forecast bike demand and simulate the evolution of the network. They refine their model using some weather measures (temperature, rainfall, wind and humidity). Etienne and Latifa (2014) model the traffic and cluster stations into work, residential, leisure, and train stations. They also show that the trip count depends on the day of week and the hour of the day. Borgnat et al. (2011) model the time evolution of Lyon's bike-sharing system using autoregressive methods. This work also analyzes the effect of connections on the user's behavior.

These works showed that the traffic in the network depends on the distance of the trip (Mahmoud et al.; Faghih-Imani and Eluru, 2016), the vicinity of the station (Mahmoud et al.; Faghih-Imani and Eluru, 2016; Hampshire and Marla, 2012; Etienne and Latifa, 2014), the neighboring stations (Rudloff and Lackner, 2014), the available connections (Faghih-Imani and Eluru, 2016; Hampshire and Marla, 2012; Etienne and Latifa, 2014; Gebhart and Noland, 2014), the weather (Mahmoud et al.; Gebhart and Noland, 2014; Borgnat et al., 2011; Vogel et al., 2011; Vogel and Mattfeld, 2011), the temperature (Mahmoud et al.; Rudloff and Lackner, 2014; Gebhart and Noland, 2014; Vogel et al., 2011; Vogel and Mattfeld, 2011), the day (Faghih-Imani and Eluru, 2016; Hampshire and Marla, 2012; Etienne and Latifa, 2014; Borgnat et al., 2011; Vogel et al., 2011; Vogel and Mattfeld, 2011), the period of the year (Rudloff and Lackner, 2014; Gebhart and Noland, 2014; Borgnat et al., 2011), the number

of bicycles in service (Borgnat et al., 2011; Vogel et al., 2011; Vogel and Mattfeld, 2011; Shu et al., 2010), the number and the size of stations (Faghih-Imani and Eluru, 2016; Borgnat et al., 2011; Shu et al., 2010), the influence of exceptional events (Fanaee-T and Gama, 2014), the labor market (Faghih-Imani and Eluru, 2016; Hampshire and Marla, 2012; Borgnat et al., 2011) and the sociology of the people (Mahmoud et al.; Faghih-Imani and Eluru, 2016; Etienne and Latifa, 2014; Borgnat et al., 2011; Vogel et al., 2011; Vogel and Mattfeld, 2011; Lathia et al., 2012). MEng (2011) proposed a method to estimate the propensity to pay for users. Lin and Yang (2011); García-Palomares et al. (2012); Lin and Chou (2012); Martinez et al. (2012) proposed also some methods to determine the optimal number of stations and their location in terms of the environment of the station, the available infrastructure and the expected demand. These studies show that the demand depends on a lot of factors, however, a few of them are pertinent in the short term forecasting demand.

2.3 Second level : Traffic analysis

Demand modelization is essential to a good redistribution model that bases its decisions on the expected demand. Even if most papers on rebalancing use a simple modelization of the demand, some papers propose more accurate modelizations. However, the stochasticity of the problem is important, and it is not completely predictable. A simpler version of the problem studied by most papers is the modelization of the global demand, i.e. the total number of departures or arrivals in the given period. Papers analyzing the demand focus on two objectives the first is to find and evaluate relations between the demand and some factors, the second tries to predict as accurately as possible the demand. The first papers showed that the demand depends on the hour, the week day, the month, the temperature and the meteorology(Borgnat et al., 2011; Gebhart and Noland, 2014; Mahmoud et al.; Vogel et al., 2011; Vogel and Mattfeld, 2010). They also prove the dependency to humidity, wind speed, night, number of stations, user gender, universities, jobs and neighborhood of the station. Bordagaray et al. (2016) identify five purposes of trips and classifies trips into these five classes : round trips, rental trips reset, bike substitution, perfectly symmetrical mobility trips and non-perfectly symmetrical mobility trips. Borgnat et al. (2011); Gebhart and Noland (2014); Hampshire and Marla (2012); Lathia et al. (2012); Mahmoud et al.; Vogel et al. (2011); Vogel and Mattfeld (2010) propose statistical models for global demand prediction. They achieve a R^2 (explained variance) score between 50 and 70%. Other authors focus their work into predicting as accurately as possible the global demand. A Kaggle (kag, 2014) contest took place in 2014-2015 to predict the demand in the Washington network. A RMLSE (root mean logarithmic squared error) score of 0.33 was achieved. The most effective

algorithms were based on a gradient boosted trees (Yin et al., 2012).

Wang (2016) analyzed the traffic of New York and tried to accurately predict the global demand using weather and time features. He analyzed the features and proposed a random forest regressor to complete missing weather data. He also improved significantly its performance by using a log transformation of the trips counts. Cagliero et al. (2017) build a model to predict critical status of stations using Bayesian classifiers, SVM, decision trees. They use the hour, the day and if it is a working day to predict if a station is critical (full or empty) Rudloff and Lackner (2014) also predict traffic per station and uses neighboring information to refine its prediction. Yin et al. (2012) predicts the global demand on Washington's network using time and weather features. They show that the prediction problem is highly nonlinear. They used a gradient boosted tree to predict the demand. Li et al. (2015) use a gradient boosted tree to predict the traffic by station, by clustering them geographically. Vogel and Mattfeld (2010); Yoon et al. (2012) used an ARMA (AutoRgressive Moving Average) model for the demand. Some deep learning models have been proposed (Polson and Sokolov, 2017). These models can compete with the gradient boosted tree but need much more tuning experience to be optimized.

Zhang et al. (2016) analyzed the trip behavior and shown a time dependency of trips. They try to predict the arrival station and time of users, knowing their departure station and time. Most papers use a one-hour time step for prediction, Rudloff and Lackner (2014) prove that this time step is a good balance between time precision and count precision. A lot of different models have been developed to explain and predict bicycle traffic, though only a few articles tried to model the traffic per station, most articles are interested in the global prediction. The traffic prediction par station problem is different from the previous one because the stochasticity is much more present. This stochasticity makes the problem more challenging. A proposed strategy to predict the traffic is to reduce the problem using clustering (group together similar stations) (Li et al., 2015; Vogel et al., 2016; Chen et al., 2016). The clustering can be performed geographically (Li et al., 2015) or only using station behavior (Vogel et al., 2016). The prediction on the clusters are understandable and accurate but the operator usually needs an estimation of traffic per station, and such models can only capture global behaviors. Vogel et al. (2016) propose simple rules to deduce from the traffic estimation between clusters, the traffic between stations. These rules give good insight on the traffic, but constraint stations too much besides underfitting the data.

Some papers also focused their work on predicting the evolution of the network in a short time period : Cagliero et al. (2017) try to predict when stations will be in a critical state. Yoon et al. (2012) build a model that predicts with high accuracy the network status in the

next hour, allowing the user to find available stations.

Rudloff and Lackner (2014) proposed a model for predicting traffic per station. The proposed model tries to predict the traffic per station and per hour using time and weather data. They also use neighboring station status as features for their algorithm. A simple linear regression is performed on the data, using categorical variables for time, weather, season, week day and temperature. Several distributions have been tested on the data and they conclude that the Poisson model is not always the best distribution and that the zero inflated and negative binomial distributions are also well suited for this problem. However, they tested their model only in Vienna's system, a medium-sized bike sharing system (about 100 stations). Their conclusions could be different on other networks. Proposed distributions increase the probability of zero trips and is then suited for a low activity network. Their study also tried to compute the influence of neighboring critical status on the demand, but they could not establish a clear dependency.

Gast et al. (2015) proposed a model to predict traffic using historical data. They used a queuing model, estimating for each time step the Poisson parameter. They used 15 minutes time steps. They estimated the Poisson parameters naively, taking the average value for all previous days. The authors also showed that the RMSE (Root mean square error) score, and similarly all precision scores are not adapted to stochastic problems, as perfect models do not give zero errors. They introduce new scores to compare models but neglected the log likelihood 4.7. They also compare their work to very naive approaches.

Yoon et al. (2012) propose a model that uses real-time network status to predict its evolution in a short horizon of time. They use a similarity approach to simplify the problem. They propose an ARMA (AutoRegressive Moving Average) model that considers neighboring stations, time and meteorology to predict the evolution of the network. The neighborhood is estimated with three different methods : A K-NN (k nearest neighbors) on station patterns, linear regression to predict the return station and a Voronoï approach to define neighboring stations. They achieve similar results with the three proposed station aggregation patterns. Their results are hardly comparable as they use short time RMSE and the precision of the algorithm. They achieve a RMSE score of 3.5 after one hour, i.e. the model is 3.5 trip away from the real value after one hour on average.

Zhou et al. (2018) build a model using Markov's chain to compute the stable point of the network and infer the out and in flow of each station. They also use weather and time feature to refine their model. However, they only predict the expected daily traffic.

2.4 Second Level : Estimating Lost Demand

The prediction of demand from historical data is intrinsically biased since it covers only satisfied demand. However, part of the demand is not satisfied, mostly due to critical situations (full or empty) in stations. This part of the demand is most of the time minor compared to the total demand but needs to be considered for the evaluation of prediction and rebalancement algorithms. The absence of lost demand estimation, leads to reproduce bad decisions by reproducing the operator's strategy. Papers proposed several ways to counter this bias, the most common is to use the random generation of trips from a Poisson distribution, whose parameters have been deduced from historical data (Brinkmann et al., 2015; Shu et al., 2010; Alvarez-Valdes et al., 2016). This strategy generates a traffic slightly different from the real one, to prevent overfitting. This bias is ignored in Vogel and Mattfeld (2010); Caggiani and Ottomanelli (2012); Mahmoud et al.; Yin et al. (2012); Schuijbroek et al. (2013). Rudloff and Lackner (2014); Chen et al. (2016) clustered stations to erase lost demand bias. Yoon et al. (2012) used similarity measured between stations to estimate lost demand.

2.5 Third level : Redistribution

The redistribution problem is one of the most addressed problems in the literature about bike sharing. The redistribution operations are meant to be the third level. However the length the time needed to solve the problem makes it more second level : These algorithms can only be used to plan the rebalancing. They cannot be used in real time. The redistribution problem can be cast as an optimization problem : the goal is to maximize the total number of trips, while minimizing the cost of redistribution. The redistribution problem is a pickup and delivery problem Raviv et al. (2013), with a fixed number of trucks and a variable and stochastic demand. The problem needs to track the time steps, the number of trucks, the number of stations and the status of the network. All these components make the problem intractable, and some approximations are necessary to solve the problem. The first approximation is to consider the static problem, which is realistic during the night. The aim of this problem is then to reach some pre-established objectives at each station, while minimizing some measure as the cost of the company or the rebalancing time. Some approximations are also made on the number of trucks, usually reduced to one, with finite or infinite capacity. Chemla (2012) proposes a solution, using two trucks and an infinite amount of time and tries to minimize the traveled distance. Benchimol et al. (2011) proposes a solution with one truck of finite capacity and an infinite time horizon, minimize the company cost. Raviv and Kolka (2013) try to minimize the lost demand, and maximize the service level, a measure

of the expected proportion of satisfied demand. Schuijbroek et al. (2013) use integer programming and constraint programming to solve the problem. They also propose a heuristic that performs better than the solution of the integer and constraint programming because of the approximations made in these models. All these formulations have the same scalability problem : they manage to solve the problem to optimality (with some approximations) only for networks with fewer than 100 stations. Some papers also try to solve the dynamic problem. This problem offers the possibility to give incentives to users to do some specific trips (Chemla, 2012; Fricker and Gast, 2013; Waserhole and Jost, 2012). They also conclude that these incentives cannot replace the operator. Chemla (2012) proposes three greedy strategies to rebalance the network and concludes that modeling the demand can considerably improve rebalancing. Contardo et al. (2012) propose a mathematical model to solve the problem, but the scalability issue is not solved, and solutions can be found only on small instances. Lin and Chou (2012); Leduc (2013); Lowalekar et al. (2017) also proposed some formulations of the problem. Lowalekar et al. (2017) uses expected future demand and scenarios, to find efficient repositioning strategies.

All these approaches try to optimize the satisfied demand, and use objectives, intervals, or demand estimation to simulate the demand. The most common strategy is to simulate the demand through a Poisson process (Alvarez-Valdes et al., 2016; Raviv and Kolka, 2013; Vogel et al., 2016; Vogel and Mattfeld, 2010). However, these models most of the time do not consider the influence of external factors. The Poisson hypothesis is pertinent in most cases as it describes independent arrival and departure on each station. Other articles use given objectives to be satisfied (Anily and Hassin, 1992; Benchimol et al., 2011; Berbeglia et al., 2007; Chemla et al., 2013; Hernández-Pérez and Salazar-González, 2004; Leduc, 2013). Usually these objectives are defined by the operator and are used to take rebalancing decision. They are defined, empirically, using historical demand. Schuijbroek et al. (2013) propose intervals based on the service level instead of targets, as objectives. This approach is more complete as it considers the variability of the demand for rebalancing and ensure a minimum service level on the network. This approach is also close to the one used by operators to rebalance the network : they use intervals to decide when to rebalance a given station. Schuijbroek et al. (2013) model the demand through a Markov chain, with some given probability to remove or add a bicycle. The modeled demand is time dependent. Caggiani and Ottomanelli (2012) propose a fuzzy logic algorithm to model the urgency of rebalancing a given station. This method also gives a heuristic to rebalance the network. However, it does not consider the distances.

2.6 Third Level : Targets for Rebalancing

Most rebalancing methods uses fill targets on each station, the same objectives used by the operator to make decisions. However, these objectives should be defined carefully because they are essential to assure a good decision process and should consider several factors such as the future demand. These objectives can be formulated as the expected demand or can have more complex forms. A lot of articles (Alvarez-Valdes et al., 2016; Raviv and Kolka, 2013; Vogel et al., 2016, 2014) suppose that the demand has a Poisson distribution, but ignore the effect of external factors as the weather and the time. Other articles suppose these objectives are given, (Anily and Hassin, 1992; Benchimol et al., 2011; Berbeglia et al., 2007; Chemla et al., 2013; Hernández-Pérez and Salazar-González, 2004; Leduc, 2013). Schuijbroeck is the first to propose intervals and not only objectives and uses them for his rebalancing algorithm. These intervals are also used by Brinkmann et al. (2015) to solve the rebalancing problem and in short and long-term heuristics. Caggiani and Ottomanelli (2012) uses a fuzzy logic algorithm to model the urgency of rebalancing on each station.

2.7 Summary

Table 2.1 and 2.2 presents a summary of the main contributions in terms of rebalancing (2.1) and traffic prediction (2.2). For each article it presents how they model each key part of the problem, and what is their objective. Table 2.1 presents some rebalancing papers. Some of these papers predict traffic, and other suppose a distribution of arrivals and departures. The table also tells how the traffic model is used in the work. Table 2.2 presents main works in terms of traffic prediction. It presents which features are used, what is the goal of the paper, which prediction level they use. And gives a quick overview of their prediction method. This work is presented in the last line of this table. The last column presents some scores presented in the papers. In this table, LR stands for linear regression, GBT for gradient boosted tree, SVR for support vector regression, SVM for support vector machine, GLM for generalized linear models, ZI for the zero inflated distribution. The NB acronym is ambiguous, if it is in the distribution column, it stays for Negative binomial. If it is in the prediction column, it stays for Naive Bayes.

Table 2.1 Contribution in bike-sharing system rebalancing

paper	target	features	distribution
Brinkmann et al. (2015)	expectation	-	-
Chemla et al. (2013)	trip data (theoretical)	-	-
Leduc (2013)	median deviation	-	-
Alvarez-Valdes et al. (2016)	expectation	time	Poisson
Raviv and Kolka (2013)	expectation	time	Poisson
Lowalekar et al. (2017)	expectation	time	-
Ghosh et al.	intervals	time	-
Schuijbroek et al. (2013)	intervals	time	Markov chain

Table 2.2 Main contribution is traffic modeling

paper	target	features	level	distribution	prediction	reduction	score
Mahmoud et al.	analysis	time, weather, environment	per station	-	linear	-	$R^2 = 0.68$
Faghih-Imani and Eluru (2016)	analysis	time, weather, environment	per station	-	linear	-	$MAE = 1.9$, $RMSE = 3.3$
Lathia et al. (2012)	analysis	time	per station	-	-	-	-
Yin et al. (2012)	expectation	time	global	-	LR,GBT,SVR	-	$RMSLE = 0.31$
Hampshire and Marla (2012)	expectation	time, weather, environment	global	-	LR	-	Barcelona $R^2 = 0.64$ Seville $R^2 = 0.22$
Yoon et al. (2012)	expectation	time, neighborhood	station	-	ARMA	-	60min :$RMSE = 3.5$
Cagliero et al. (2017)	critical status	time	station	-	NB,SVM,	-	60min :$F1$=0.72 120min :$F1$=0.67
Gebhart and Noland (2014)	expectation	time, weather, environment	global	Negative binom	linear	-	$R^2 = 0.2$
Borgnat et al. (2011)	expectation	time, weather	global	-	linear	cluster	-
Nair et al. (2013)	distribution	time	per station	NB, Poisson	linear	-	prob of empty improved by 0.05%
Rudloff and Lackner (2014)	distribution	time, weather	per station	Poisson, NB, ZI	GLM	-	deviation of 3% per day
Chen et al. (2016)	expectation	time, weather	per cluster	Poisson	linear	cluster	$F1 = 0.9$
Vogel et al. (2011)	analysis	time, weather	per cluster	-	-	cluster	-
Vogel et al. (2016)	generation of trips	time	per station	-	LR	cluster	-
Li et al. (2015)	expectation	time, weather	per cluster	-	GBT	cluster	$RMSLE = 0.4$ $MAE = 0.35$
Gast et al. (2015)	modelization of traffic	hour	per station	Poisson	mean	-	$RMSE \in [2, 8]$
This work	distribution	time, weather	per station	Poisson, NB, ZI	GBT/RF	SVD/kmeans	-

This work builds a second level traffic model for rebalancing purposes (Brinkmann et al., 2015; Chemla et al., 2013; Leduc, 2013; Alvarez-Valdes et al., 2016; Raviv and Kolka, 2013). This work tries to propose a detailed understanding of the network behavior using time and weather features and predicts traffic per station. The model predicts the statistical distribution of trips, allowing a finer use of the results (compute the service level). This work differs from the work of Nair et al. (2013); Rudloff and Lackner (2014); Vogel et al. (2016); Li et al. (2015) by reducing the problem to achieve better performance wisely. A combined approach is proposed to improve the results. Several scores are computed to compare as precisely as possible the different approaches. Finally, this model is applied to the Bixi system to improve the rebalancing strategy. Several scores are proposed to evaluate the contribution of the model.

CHAPTER 3 DATA ANALYSIS

The first step to build a machine learning model is to analyze the data, clean it, preprocess it and select useful factors. This chapter presents all the data used during this work. It is then analyzed to remove irrelevant data and correct it. This work is based on Montreal data, data from New York and Washington, used for validation, is very similar.

3.1 Bixi's Data

First, we look at Bixi data. Bixi is a third-generation bike-sharing system. As most of these systems they track in real time the state of stations and bikes regarding their rentals and returns. Bixi has also developed an application that tracks the position and actions of rebalancing trucks. Like most bike-sharing systems Bixi makes part of its data publicly available. These data are composed of :

— the history of all trips, per month and per year

— the status of stations in real time.

Trip history Bixi makes public all its trip history since its opening in 2014. This work uses data from 2015 and 2016 and covers 10 thousand hours of service. This data is composed for each trip of its start and end station, its start and end time, and the trip duration. The total duration of the trip in milliseconds is also available. These trips are gathered in a csv file and grouped per month (one file per month). The data used in this work was composed of 7 million rows. Montreal network is closed during the winter then there are no trip data during this period. The data used goes from the 15th of April 2015 to the 15th of November 2015 and from the 15th of April 2016 to the 15th of November 2016.

Real-time station status We used for this station data composed by the station position, its capacity, its name and its id. This information is supposed static, even if the operator can modify it during the year (capacity, location and availability). Relocations and shutdowns are always temporary (city work). Bixi tries to place stations always in the same area (a maximum displacement of one city block). Station capacity can change along the year, but only the current capacity is needed in this work. Bixi makes current station information publicly available, they moved it to the GBFS format, as Citybike (New York) and capital Bikeshare (Washington). The GBFS norm definition is available at `https://github.com/NABSA/gbfs`. Bixi's data is accessible on the following link `https://api-core.Bixi.com/gbfs/gbfs.json`

	A	B	C	D	E	F	G	H
1	Start date	Start station r	Start station	End date	End station nu	End station	Account type	Total duration (ms)
2	30/04/2015 23:59	6926	Marie-anne/ S	01/05/2015 00:07	6200	Maguire / Sair	Member	482182
3	30/04/2015 23:59	6143	Rachel / Brébe	01/05/2015 00:09	6236	Bordeaux/La	Member	620335
4	30/04/2015 23:59	6177	Saint-Hubert /	01/05/2015 00:07	6274	De la Roche/S	Member	495310
5	30/04/2015 23:59	6177	Saint-Hubert /	01/05/2015 00:22	6374	15e Avenue /	Member	1369031
6	30/04/2015 23:59	6729	St-André/Mai	01/05/2015 00:15	6196	Villeneuve/St-	Member	968004
7	30/04/2015 23:59	6193	Esplanade/Fa	01/05/2015 00:11	6212	Duluth/Esplan	Member	670522
8	30/04/2015 23:58	6218	Prince-Arthur	01/05/2015 00:01	6214	Square Saint-L	Casual	157980
9	30/04/2015 23:58	6039	McGill / des R	01/05/2015 00:06	6089	Saint-Jacques	Member	460031
10	30/04/2015 23:58	6394	Valois / Ontar	01/05/2015 00:02	6387	Métro Joliette	Member	264218
11	30/04/2015 23:58	6049	King / Welling	01/05/2015 00:16	6427	Rushbrooke /	Member	1042929
12	30/04/2015 23:58	6149	Chapleau / M	01/05/2015 00:10	6391	Aylwin / Onta	Member	700412
13	30/04/2015 23:58	6169	St-André/Laur	01/05/2015 00:02	6094	Gaspé / Fairm	Member	223074
14	30/04/2015 23:59	6261	Louis Hémon	01/05/2015 00:13	6329	Gounod / Sair	Member	841555
15	30/04/2015 23:57	6383	Jardin Botanic	01/05/2015 00:01	6417	Desjardins/Hc	Member	279598
16	30/04/2015 23:57	6034	Saint-Urbain /	01/05/2015 00:29	6182	Bullion / Mon	Casual	1926213
17	30/04/2015 23:57	6729	St-André/Mai	01/05/2015 00:08	6138	Gauthier / De	Member	696495
18	30/04/2015 23:57	6277	Louis-Hémon	01/05/2015 00:01	6268	Chambord / B	Member	253085
19	30/04/2015 23:58	6178	Rivard /Rache	01/05/2015 00:03	6211	Roy / Saint-La	Casual	293569
20	30/04/2015 23:57	6205	Milton / du Pa	01/05/2015 00:08	6044	Roy/St-Huber	Member	653102
21	30/04/2015 23:58	6901	Sainte-Cather	01/05/2015 00:05	6114	Métro Papine	Member	458175
22	30/04/2015 23:56	6088	Guy / Notre-D	01/05/2015 00:06	6168	Hotel de ville/	Member	618302
23	30/04/2015 23:56	6215	Saint-Cuthber	30/04/2015 23:57	6903	Napoleon/St-	Member	59449
24	30/04/2015 23:56	6215	Saint-Cuthber	01/05/2015 00:02	6203	Hutchison / St	Casual	348309
25	30/04/2015 23:55	6386	Métro Préfon	01/05/2015 00:00	6391	Aylwin / Onta	Casual	306944
26	30/04/2015 23:55	6386	Métro Préfon	01/05/2015 00:00	6391	Aylwin / Onta	Member	340923
27	30/04/2015 23:55	6426	Willibord / Ve	30/04/2015 23:56	6309	5e ave/Verdu	Member	92432
28	30/04/2015 23:54	6205	Milton / du Pa	01/05/2015 00:06	6201	Villeneuve / d	Member	739631
29	30/04/2015 23:53	6235	Saint-Dominic	01/05/2015 00:06	6368	10e Avenue /	Member	824039

Figure 3.1 Example of a trip file

This format is composed of two data sets : The immutable information and the mutable information. The first one gathers information about the station properties (location, capacity, ...) the second one gathers the station status. The first data set, accessible at `https://api-core.Bixi.com/gbfs/en/station_information.json` presents :

- the id of the station
- the complete name of the station
- its short name
- the latitude
- the longitude
- the rental payment methods
- the capacity

Example :


```
{
"station_id":"1",
"name":"Gosford / St-Antoine",
"short_name":"6001",
"lat":45.50981958364862,
"lon":-73.55496143951314,
```

```
"rental_methods":["KEY","CREDITCARD"],
"capacity":19,
"eightd_has_key_dispenser":false
}
```


The second data set is accessible at `https://api-core.Bixi.com/gbfs/en/station_status.json` and presents the station status, i.e. :

- station ID
- number of available bikes
- number of unavailable bikes (blocked, or damaged)
- number of available docks
- number of unavailable docks (off or damaged)
- a Boolean that is true if the station is installed
- a Boolean that indicates if rentals are allowed
- a Boolean that indicates if returns are allowed
- last connection timestamp


```
{
"station_id":"1",
"num_bikes_available":0,
"num_bikes_disabled":0,
"num_docks_available":2,
"num_docks_disabled":17,
"is_installed":0,
"is_renting":0,
"is_returning":1,
"last_reported":1480007079,
"eightd_has_available_keys":false
}
```


3.1.1 Bixi private data

Bixi has also access to other data, not available online. Bixi collects information about its infrastructure : stations, docks, bikes, and keys and about its actors : technicians, clients, customer service agents and mechanics. Bixi works with several external companies that manage their data and help them understanding the network status.

3.2 Weather data

	Temp. °C	Point de rosée °C	Hum. rel. %	Dir. du vent 10's deg	Vit. du vent km/h	Visibilité km	Pression à la station kPa	Hmdx	Refr. éolien	Météo
HEURE										
00:00	17,0	10,8	67	26	8	24,1	101,26			ND
01:00	16,7	11,1	69	23	9	24,1	101,25			Généralement dégagé
02:00	15,7	9,8	68	24	11	24,1	101,21			ND
03:00	15,6	10,0	69	22	10	24,1	101,20			ND
04:00	15,5	11,5	77	31	7	24,1	101,22			Généralement dégagé
05:00	14,8	11,7	82	22	9	48,3	101,24			ND
06:00	16,1	11,9	76	26	14	48,3	101,23			ND
07:00	17,4	12,2	72	21	7	48,3	101,30			Généralement dégagé
08:00	18,6	12,7	68	26	10	48,3	101,30			ND
09:00	20,1	12,8	62	21	7	48,3	101,31			ND
10:00	21,4	14,6	65	22	6	48,3	101,26	25		Généralement dégagé
11:00	21,9	14,2	61	19	17	48,3	101,24	25		ND
12:00	23,5	15,1	59	20	16	48,3	101,19	28		ND

Figure 3.2 Example of precipitation file

The weather information is taken from *données Canada* (don, 2018) and provides data with an hourly precision. The data is composed of quantified measures as visibility, humidity, atmospheric pressure, temperatures and qualitative measures for the weather (rain, snow, fog, ...). The meteorological data come from *Montreal INTL A* station (id 7025251). The data used for this work comprehends the period between 01/01/2015 and 31/12/2016.
The data is presented in a csv file with :

- the date and hour
- a quality indicator of the data
- the temperature (°C)
- the dew point
- the relative humidity (%)
- the wind direction (per 10 deg)
- the wind speed (km/h)
- the visibility (km)
- the pressure (kPa)
- the wind chill

— the weather (text)

Figure 3.2 shows an example of weather data. We can see that the weather column has a lot of missing data. Next paragraph covers this problem.

Remark : The weather data used in this work is the observed data. However, the model we are going to build is the next chapter is intended to predict future demand and thus should use weather prediction. However, we chose to learn our model on the observed weather. Firstly, because the demand reacts to the real weather and not to the predicted one, secondly because weather predictions three to six hours in advance are quite precise and lastly because learning the model on the predicted weather introduce even more stochasticity in the problem.

3.3 Holidays and National Days

We collected Montreal's holidays and national days manually on the internet. They were added as two new features, indicating whereas an hour is during a holiday or national day. We differentiate public holidays called *national days* in this paper and school holidays called *holidays*.

3.4 Preprocessing

Once the data is collected, it has to be preprocessed. This step aims to shape the data removing gaps and inconsistent values.

3.4.1 Features preprocessing

First we build a feature matrix from the weather and time data. We augment it with holidays and national days. The following operations are computed :

1. Missing values in the weather data are filled with the nearest available value (a maximum distance of five hours is authorized). Missing data was found for *weather*, *wind speed*, *temperature*, *visibility*, *pressure*, *humidity* and *wind chill*. More than half of *wind chill* data is absent, this feature is then not considered in the rest of this work. This missing data completion operation is detailed in Section 3.5.
2. Textual weather is transformed into categories : *rain*, *showers*, *snow*, *ice*, *drizzle*, *fog*, *cloudy*, *storm*, *heavy*, *moderate* and *clear*.
3. The date time column is split into five columns : *year*, *month*, *day*, *hour* and *day of the week*. Due to the evident non-linearity dependence between the traffic and the hour of

the day, categorical variables are introduced for each hour (24 new variables).

4. The plot of trip counts per hour of the week (Figure 3.3) shows three different behaviors, one for the weekend, one for Monday and Friday and one for the remaining days of the week. The literature on bikesharing use categorical variables for week days. They create two (Yoon et al., 2012; Yin et al., 2012; Lathia et al., 2012), three (Rudloff and Lackner, 2014) or seven (Vogel et al., 2016; Borgnat et al., 2011; Li et al., 2015) categorical variables. In this work we choose to use three categorical variables. Then three weeks days categorical variables are created. The first regroups Sunday and Saturday, the second Monday and Friday and the last, Tuesday, Wednesday and Thursday. Days in each category are very similar, then using seven categories instead of three is not interesting as it adds a lot of parameters for a small potential precision improvement.
5. A *holiday* and a *National day* columns are added to the data.

This data in represented using a matrix representing for each hour (row) its features (columns). This matrix is used to learn the model.

3.4.2 Objective preprocessing

The trip data is also transformed to get the objective matrix. In this work we model arrivals and departures in a station independently. We suppose that the behavior of arrivals and departures are not related. Therefore we consider them separately. We push this hypothesis even further considering that arrivals and departures are equivalent. This hypothesis supposes that arrivals and departures can be modeled using the same approach. Hence stations are separated into two sub stations, one that counts only arrivals and one that counts only departures. Until Chapter 5, we split stations in two new stations, one that count departures

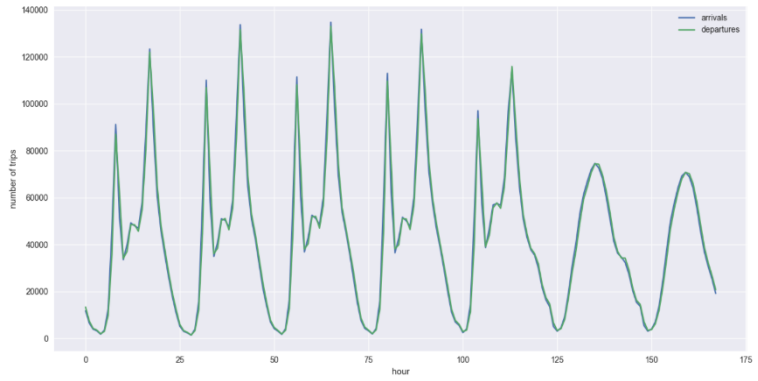

Figure 3.3 Repartition of traffic during the week (Monday to Sunday)

and one that count arrivals. Complete stations are recomposed in the last chapter to build decision intervals, and model the behavior of a station fills level.

To preprocess trip data, we first aggregated it by stations and hour, in a simple matrix. Then if there are n stations in the network, there are $2 \times n$ columns in the matrix. The objective matrix represents for each hour, the number of arrivals and the number of departures of each station. This column is important because the trip data is not complete : the network is closed during winter. Hence, these hours are excluded from the learning data.

To build this matrix, we choose a one-hour time step. This choice is justified by Rudloff and Lackner (2014) whom proved that it is a good compromise between stochasticity and precision. This time step is also practical, because features are given per hour. This precision is also sufficient for the operator that needs long time prediction up to $7 \times 24 = 168$ hours.

We also tried to transform data to improve the performance of algorithms. Section 4.5.1 presents some data transformation as normalization, log transformation (Wang, 2016) and features decorrelation.

3.5 Missing Data

The analysis of weather data showed that there were some missing data. Table 3.1 shows the number of missing values per data column. We found 13 missing hours for all data, probable to station shut down. However, none of these 13 hours were consecutive, hence we filled them using a forward fill method (fill with the data of the previous hour). This approximation is reasonable because of the small variations of meteorological measures between two consecutive hours. This estimation neglects possible, but unlikely, exceptional events during the hour. The more problematic point is the number of missing weather values : more than half of the values are missing, then a more accurate analysis is needed.

Table 3.1 Missing data numbers

column	data missing	% data missing
temperature	6	0.05
Humidity	6	0.05
visibility	6	0.05
wind	6	0.05
pressure	6	0.05
wind chill	10133	99
weather	6115	59

To analyze weather missing values we computed the gap sizes (number of consecutive missing values) on the whole weather dataset (train + test). The results are reported in table 3.2. It appears that most gaps are of size 2. The study of the data also shows that most weather values do not vary between three consecutive hours (identical for two gaps over three) i.e. between the hour before and after the gaps. Hence completing the missing values using the nearest available data is justified. This data seems to be recorded every two hours and not every hour.

Table 3.2 gap occurrences

gap sizes	1	2	3	4	5
occurrences	225	2941	1	0	1

3.6 Feature Analysis

In the previous subsections we built the features of table 3.3. In this subsection, we select the significant features.

Table 3.3 Available features for the model

Time features	Weather features
Year (integer)	Temperature (continuous)
Month (integer $\in [1, .., 12]$)	Visibility (continuous)
Day of month (integer $\in [1, .., 31]$)	Humidity (continuous)
Day of week (integer $\in [1, .., 7]$)	Pressure (continuous)
Hour (integer $\in [0, .., 23]$)	Wind (continuous)
Monday/Friday (binary)	Rain (binary)
Tuesday/Wednesday/Thursday (binary)	Cloudy (binary)
Saturday/Sunday (binary)	Brr (binary)
National day (binary)	Showers (binary)
Holiday (binary)	Fog (binary)
Hour 0 (binary)	Drizzle (binary)
...	Storm (binary)
Hour 23 (binary)	precip (binary)
	Heavy (binary)
	Moderate (binary)
	Snow (binary)
	Ice (binary)
	Clear (binary)

The Figure 3.4 shows the dependency between the number of trips in one hour and the

features. To build this plot, data is split into categories (for continuous features) and for each category we display its means and its two quartiles. There is one plot per considered feature. The first graphs show a clear dependency between features and trip number. In the last graphs, this dependency is less clear. However the absence of dependency in these graphs does not prove that these features should not be considered. The week-day feature for example is not significant, however its usage is essential to achieve good performances as user behavior changes (distribution of traffic during the day) between week days and week ends. However, these graphs are not a dependency proof. Section 3.6.1 analyze more rigorously the dependencies.

To build a good model is it also important to check the correlation between features as they may cause overfitting, or can influence the interpretation of results. Figure 3.5 represents the correlation matrix between the features. High correlation is shown in red (≈ 1), null correlations in white (≈ 0) and negative correlation in blue (≈ -1). The notable correlations between variables are :
Positively correlated features :

- *pressure* and *year*
- *precipitation* and *rain*
- *showers* and *rain*
- *showers* and *precipitation*
- *week day* and the categorical variable *Saturday/Sunday*
- *temperature* and *holidays*
- *drizzle* and *fog*
- *fog* and *brr*
- *fog* and *drizzle*

Negatively correlated features :

- the categorical variable *Tuesday/Wednesday/Thursday* and *week day*
- *visibility* and *humidity*
- *precip* and *cloudy*
- *visibility* and *rain*
- *visibility* and *precip*
- *temperature* and *humidity*

We can also point out the dependence between *hour of the day* and *temperature* and between *humidity* and *visibility*. Most of these correlations can be simply explained by dummy variables or simple observations. For example, *temperature* and *holidays* are correlated because *holidays* mostly happen during hot months, also *cloudy* and *rain* are negatively correlated because when it rains, the *cloudy* is implied and then not written. These kinds of explanations are valid for most correlated variables. However, they cannot explain the correlation between year and pressure.

This analysis shows some features that add more confusion than they explain the objective. It appears that *holidays* and *cloudy* are not highly correlated to the trip number, while being highly correlated to other features. These features are then discarded. Some variables are redundant by construction (dummy variables). Then one possibility between the dummy variables and the continuous one is chosen depending on the model. Even if some strong correlations are still present, we chose to keep most variables to improve the performance, as we try to accurately predict the traffic. Regularizations methods (penalization of big weights) will be used to limit overfitting with correlated variables.

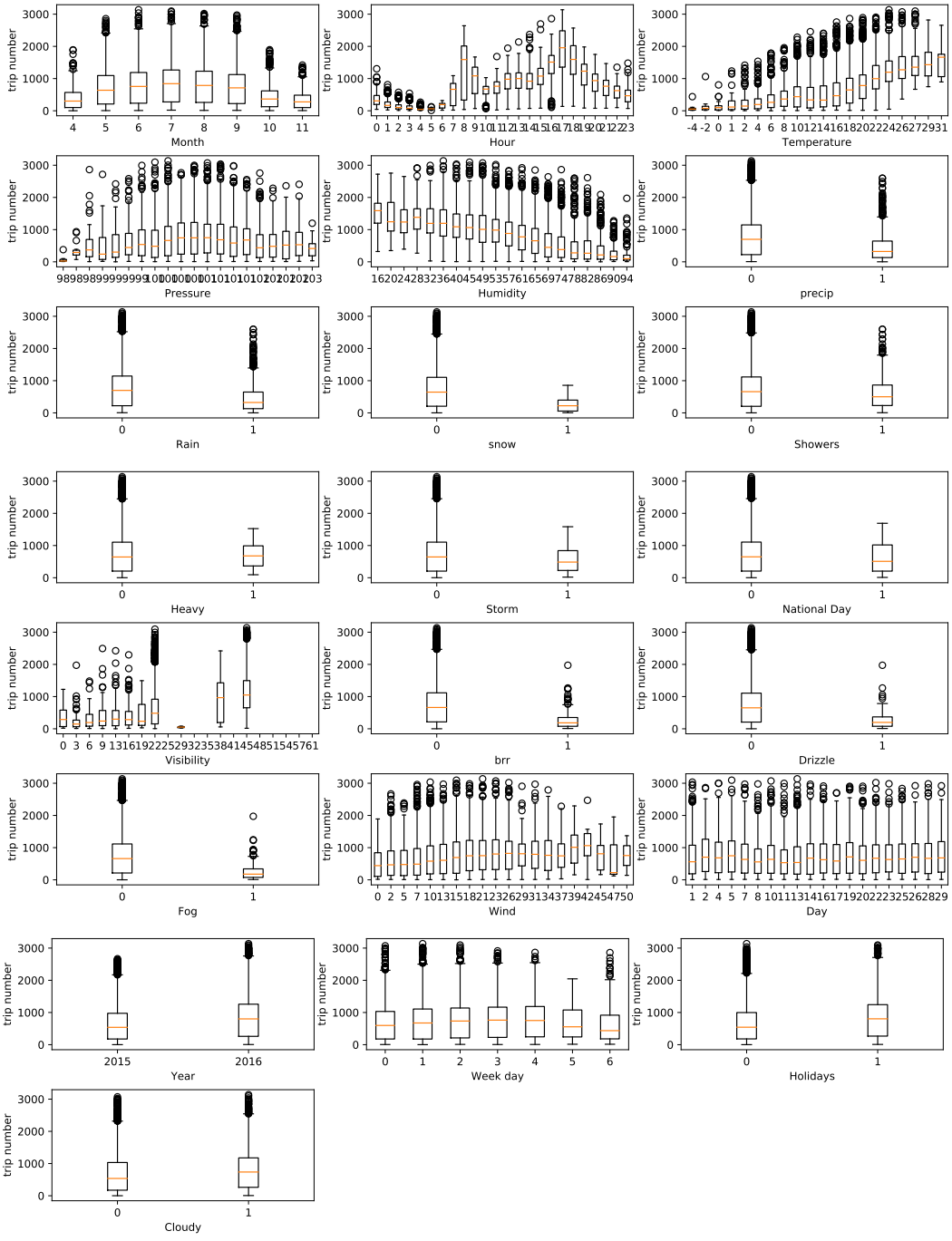

Figure 3.4 Trip Numbers per Feature

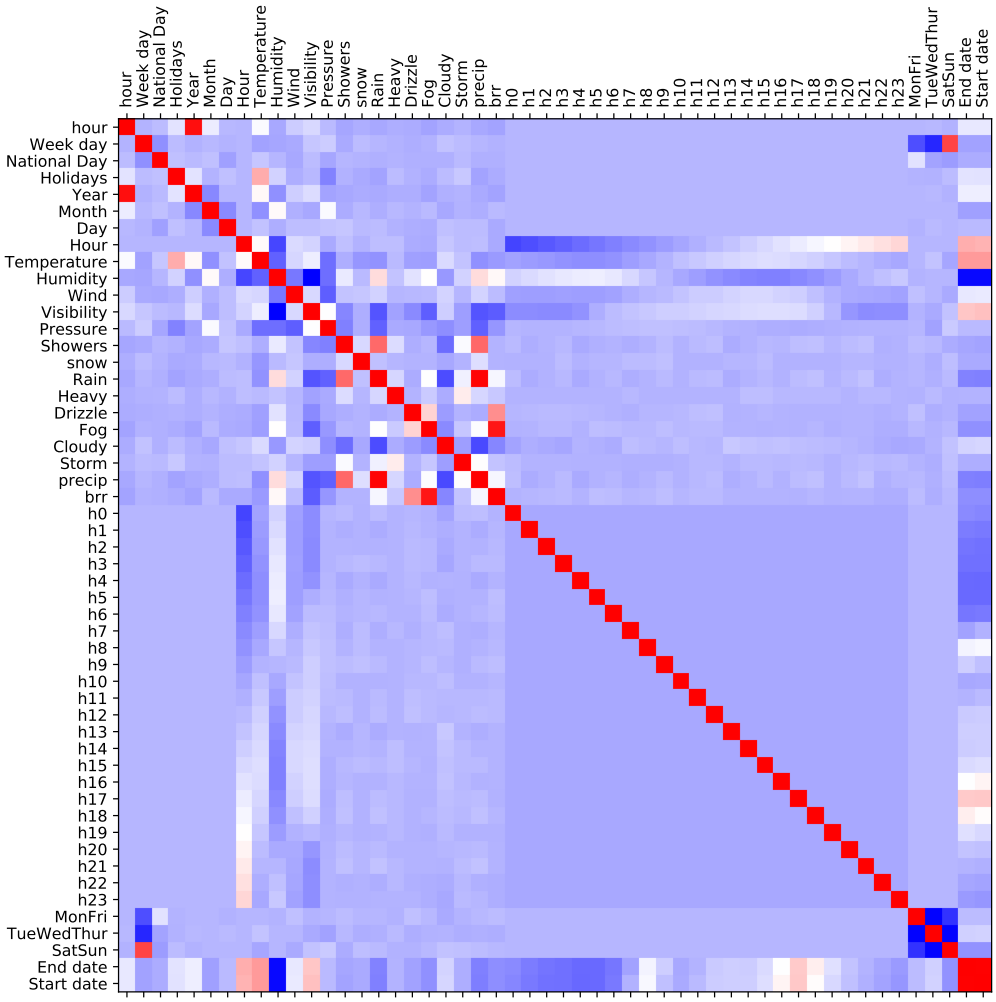

Figure 3.5 Correlation matrix generated using *pandas*, a python library

3.6.1 Feature significance

Dep. Variable:	ch	R-squared:	0.539
Model:	OLS	Adj. R-squared:	0.538
Method:	Least Squares	F-statistic:	513.6
Date:	Fri, 20 Oct 2017	Prob (F-statistic):	0.00
Time:	13:15:55	Log-Likelihood:	-65879.
No. Observations:	8810	AIC:	1.318e+05
Df Residuals:	8789	BIC:	1.319e+05
Df Model:	20		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	72.7541	428.480	0.170	0.865	-767.167	912.676
wday	-20.9798	2.328	-9.011	0.000	-25.544	-16.416
ferie	-223.8290	27.156	-8.242	0.000	-277.061	-170.597
vac	-25.9545	11.187	-2.320	0.020	-47.883	-4.026
annee	-0.6881	0.454	-1.515	0.130	-1.578	0.202
Mois	15.4354	2.775	5.562	0.000	9.995	20.875
Jour	-0.7012	0.528	-1.328	0.184	-1.736	0.333
Heure	24.8448	0.735	33.787	0.000	23.403	26.286
temp	36.5271	0.868	42.103	0.000	34.826	38.228
Hum	-9.4798	0.399	-23.783	0.000	-10.261	-8.698
vent	0.9548	0.613	1.557	0.119	-0.247	2.157
visi	8.8581	0.476	18.605	0.000	7.925	9.791
pression	14.4062	7.929	1.817	0.069	-1.136	29.949
averses	57.4573	28.242	2.034	0.042	2.097	112.817
neige	188.8432	105.631	1.788	0.074	-18.218	395.904
pluie	7.8208	23.370	0.335	0.738	-37.990	53.631
fort	113.6235	135.123	0.841	0.400	-151.250	378.497
bruine	156.5212	57.096	2.741	0.006	44.600	268.443
brouillard	86.5113	36.767	2.353	0.019	14.439	158.583
nuageux	19.1655	10.106	1.896	0.058	-0.644	38.975
orage	-97.0238	60.629	-1.600	0.110	-215.870	21.823

Omnibus:	2037.851	Durbin-Watson:	0.724
Prob(Omnibus):	0.000	Jarque-Bera (JB):	4525.623
Skew:	1.324	Prob(JB):	0.00
Kurtosis:	5.306	Cond. No.	1.90e+05

Figure 3.6 ANOVA results, computed using *scipy* OLS

To test the feature significance, we used the ANOVA (ANalysis Of VAriance) test. Results are shown in Figure 3.6. This figure shows that, with a 95% confidence (p-value at 0.05, column $P > |t|$) the number of trips depends on the *week day*, the *national days*, the *holidays*, the *month*, the *hour*, the *temperature*, the *humidity*, the *visibility*, the *showers*, the *snow*, the *drizzle* and the *fog*. The other dependencies were not proven by ANOVA. Therefore the *year*, the *day*, *wind*, *pressure*, *rain* and *heavy* (rain) dependencies, were not significant. However, this analysis tracks only linear dependency and is not able to check more complex ones. Furthermore this analysis has been done on the total number of trips and these features may

be significant for some stations. The correlation between features also makes some features not significant as their influence has already been modeled by the other feature.

This analysis also gives a R^2 score of 0.540 to model the total number of trips. This score means that 54% of the variance of trips counts have been explained by the linear regression. This score, gives a first base line for our model. This R^2 score is quite low but usual for human behavior studies. This score is computed on the learning data and may be overestimated.

3.6.2 Data analysis conclusions

In this section we completed missing data, selected the features for training and preprocessed the trip data. Table 3.4 presents the features built and validated in this section, with their type. Some of these features are redundant (Hour and Hour i). They represent two options for the learning, only one is selected for each model.

Table 3.4 Selected features for the model

Time features	Weather features
Year (int)	Temperature (float)
Month (int $\in [1, .., 12]$)	Visibility (float)
Day of month (int $\in [1, .., 31]$)	Humidity (float)
Day of week (int $\in [1, .., 7]$)	Pressure (float)
Hour (int $\in [0, .., 23]$)	Wind (float)
Monday/Friday (boolean)	Rain (boolean)
Tuesday/Wednesday/Thursday (boolean)	Cloudy (boolean)
Saturday/Sunday (boolean)	Heavy (boolean)
National day (boolean)	Showers (boolean)
Holiday (boolean)	Fog (boolean)
Hour 0 (boolean)	Drizzle (boolean)
...	Storm (boolean)
Hour 23 (boolean)	

CHAPTER 4 MODELING DEMAND PER STATION

This chapter presents the construction of the traffic model, and all the choices made to achieve a better performance. The model built in this chapter aims to estimate the probability distribution of trip counts for each station at each hour of the day, using time and weather features. This chapter aims to implement a model that predicts the traffic per station. This new problem introduces new difficulties related to the increasing stochasticity of the demand due to the large number of stations and the exploding size of the model. This chapter builds a general model able to predict the traffic efficiently for each station.

4.1 General Overview

In this problem we are trying to model a stochastic process that takes its values in $\mathbb{N}^{n\_stations}$. Several approaches can be used for this task. The **Machine Learning** approach tries to predict from features properties, with the highest precision, the number of trips. This is usually achieved minimizing the squared error. This error corresponds to the trip count expectation if the distribution of errors is Gaussian. Therefore using the prediction of the algorithm as the trip count expectation suppose a Gaussian distribution of errors. The **Statistical approach** tries to fit the distribution corresponding to the data. This approach is more focused on the distribution of data and therefore seeks to minimize the variability of the results. Hence simple predictors are used as they allow to control and understand the variance of results.

Denote by t the current time features, w the current weather features, and X_s the random variable that count the number of trips during the next hour at station s. Then this work estimates $P(X_s = k|T = t, W = w)$ the probability that the number of trips at station s is k, knowing that the current time is t and that the current weather is w. To simplify the problem, we suppose that this probability distribution follows a standard probability distribution and that this distribution can be defined by its mean and variance. Let's note $d(\mu, \sigma)$ the density function of that distribution with mean μ and variance σ, $\hat{\mu}_s(t, w)$ an estimation of X_s's mean and $\hat{\sigma}_s(t, w)$ an estimation of X_s's variance, knowing the time t and the weather w. Then we approximate the trip count probability by :

$$P(X_s = k|T = t, W = w) \simeq d(\hat{\mu}_s(t, w), \hat{\sigma}_s(t, w))(k)$$

This work focus on determining the best approximations for the three functions $d(\mu, \sigma)$,

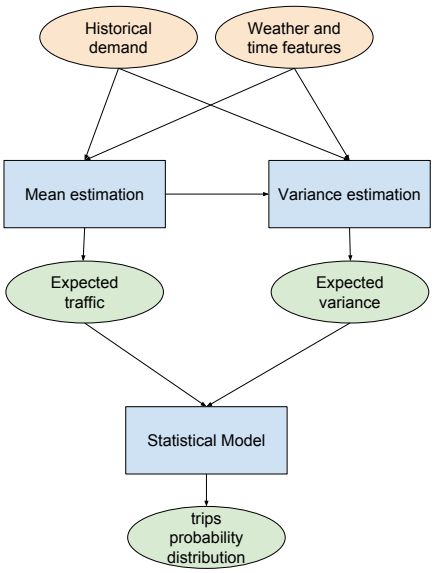

Figure 4.1 Model preview

$\hat{\mu}_s(t,w)$ and $\hat{\sigma}_s(t,w)$. Two machine learning algorithms are learned to estimate respectively $\hat{\mu}_s(t,w)$, $\hat{\sigma}_s(t,w)$. And several distribution hypotheses are tested to determine the best $d(\mu,\sigma)$ function.

Figure 4.1 shows how our model works : from historical demand and weather, the mean $\hat{\mu}_s(t,w)$ and variance $\hat{\sigma}_s(t,w)$ are estimated, then a distribution $d(\mu,\sigma)$ is fitted on each station, using the estimation of μ and σ

The goal of this chapter is to propose a new approach for predicting the traffic on each station, taking advantage of their similarities, while staying specific to each one of them. This work uses this idea to achieve state-of-the-art performance on traffic prediction of bike-sharing systems, greatly reducing its complexity. This model is essential to propose a new rebalancing strategy for Bixi (Chapter 5).

The model is divided into three parts, the three function to approximate. First this chapter builds the mean predictor $\hat{\mu}_s(t,w)$, then the variance predictor $(\hat{\sigma}_s(t,w))$ and finally determines the best distribution hypothesis $(d(\mu,\sigma))$.
All tests, models and operations, have been run on a 64-bit Windows 10 machine with :

— Intel Core i7-6700HQ CPU @ 2.60GHz, 2592 MHz, 4 core, 8 processors

— 16Go RAM

— GPU Nvidia Gforce GTX 960M

4.2 Evaluating models

This section presents scores used in this work to compare algorithms, it provides an insight on the specificity of each score, and what they represent. A lot of measures exist to compare models and each of them focus on different aspects providing different insights on the goodness of an algorithm. These measures can be computed by station or globally, giving different levels of understanding. To compare models, we first compare the global score to select the best algorithms, then we will focus on these scores per station to get a deeper understanding of the model and its limits.

The first scores are precision scores, that value how close the prediction was to reality. However, these scores have to be nuanced as rental and return demands on each station are stochastic. Then a precision score may not be suited. Gast et al. (2015) show that even if we knew perfectly the probabilistic model, the precision score will be significantly different from zero. Therefore these measures are not adapted to stochastic processes. However, they are pertinent to compare expectation estimations. Section 4.5.1 provides these scores with the log likelihood, which allow us to compare different models.

Notations :

S	the set of stations
n_s	the number of stations $n_s = \|S\|$
T	the set of hours of the dataset.
n_t	the number of hours $n_t = \|T\|$
D	is the $\|T\| \times \|S\|$ trip count matrix
$D_{t,s}$	the real value of the objective (trip count) at time t in station s
$\hat{D}_{t,s}$	the estimated value of $D_{t,s}$
$\bar{D}$	the mean of $D_{i,s}$ over time and stations. $\bar{D} = \frac{1}{\|T\|\|S\|} \sum_{t \in T} \sum_{s \in S} D_{t,s}$
C	is the set of features.
n_C	is the number of features $n_C = \|C\|$
F	is the feature matrix (time and weather features for each time step). F is a $n_t \times n_C$ matrix.

Throughout this thesis, if M is a matrix, $M_{i,j}$ is the element of M at the i^{th} row and j^{th} column. If i or j is replaced with an underscore (_) this means that the whole row or column is selected. Hence $M_{\_,j}$ is the j^{th} column of M and $M_{i,\_}$ is the i^{th} row of M. A $\bar{bar}$ indicates the means of all selected elements, a $\hat{hat}$ represent an estimation.

4.2.1 Splitting Data

The trip and feature data represent time dependent data. The trip data depends on the features but also on the activities near the station. These activities are considered constant, as they should not change during the year and then their influence is learned implicitly by the model. However changes can occur and affect the performance of the model, since the model itself is not able to automatically learn the changes in activities due to the unavailability of the data. . This process must be considered in the data split. Selecting randomly test data into the trip data gives to the learning algorithm information about the future behavior of the network, which results in an overestimation of the performance of the model. Then we choose to select the first part of the data for the training, the second part for validation and the last one for test. The training set refers to the data from 01/01/2015 to 30/06/2016, the validation set from 01/07/2016 to 31/08/2016 and the test set from 01/09/2016 to 30/10/2016.

The data of 2017 is not used because of the modification of the network made between 2016 and 2017 and the lack of yearly data for the new network. This loss is compensated by proposing some solutions to add new stations in the network (subSection 4.8.4).

4.2.2 Size of Stations

The size of a station is defined as the average number of trips per hour in the station. This number is not an error measure, but a characteristic of a station. It is computed as :

$$Size(s) = \frac{1}{|T|} \sum_{t \in T} D_{t,s}$$

We categorize stations into three categories, i.e. big stations (more than 2 trips per hour), small stations (less than 0.5 trips per hours), whereas the remaining stations are considered medium stations. Big stations represent 30% of the stations, small stations 15% and medium stations 55%.

4.2.3 MAE

The MAE or Mean Absolute Error is defined as the mean of the absolute error values. It is one of the simplest error measures. It is used in our work to better approximate the expected mean.

$$MAE(\{D_{i,s}\}_{i \in T, s \in S}, \{\hat{D}_{i,s}\}_{i \in T, s \in S}) = \frac{1}{|T||S|} \sum_{i \in T} \sum_{s \in S} |\hat{D}_{i,s} - D_{i,s}|$$

The penalization of errors is linear, hence big errors are not discouraged (as much as in other scores)

4.2.4 RMSE

The RMSE or square Root Mean Squared Error is the Root of the means of the errors squared. It is one of the most commonly used error measures. This error penalizes a large error value. It is widely used in machine learning because of its relation to the Gaussian distribution.

$$RMSE(\{D_{i,s}\}_{i\in T,s\in S},\{\hat{D}_{i,s}\}_{i\in T,s\in S}) = \sqrt{\frac{1}{|T||S|}\sum_{i\in T}\sum_{s\in S}(\hat{D}_{i,s}-D_{i,s})^2}$$

4.2.5 MAPE

The MAPE or Mean Absolute Percentage Error is a measure that quantifies the relative error. The error is computed in terms of percentages and not in terms of raw error. This error compensates the magnitude of the error which increases when the predicted value increases. The formula to compute the MAPE score is :

$$MAPE(\{D_{i,s}\}_{i\in T,s\in S},\{\hat{D}_{i,s}\}_{i\in T,s\in S}) = \frac{1}{|T||S|}\sum_{i\in T}\sum_{s\in S}\left|\frac{\hat{D}_{i,s}-D_{i,s}}{D_{i,s}+1}\right|$$

4.2.6 RMSLE

The RMSLE or square Root Mean Squared Logarithmic Error is the square root of the means of the square of the logarithm of the error. This error penalizes more positive errors than negative ones, and penalizes more large errors than small ones. It is used in cases where the prediction has to be positive.

$$\begin{aligned} RMSLE(\{D_{i,s}\}_{i\in T,s\in S},\{\hat{D}_{i,s}\}_{i\in T,s\in S}) &= \sqrt{\frac{1}{|T||S|}\sum_{i\in T}\sum_{s\in S}(log(\hat{D}_{i,s}+1)-log(D_{i,s}+1))^2} \\ &= \sqrt{\frac{1}{|T||S|}\sum_{i\in T}\sum_{s\in S}\Big(log\Big(\frac{\hat{D}_{i,s}+1}{D_{i,s}+1}\Big)\Big)^2} \end{aligned}$$

4.2.7 R^2 Score

The R squared coefficient is a measure of the deviation from the mean explained by the model.

Let us define $SS_{tot} = \sum_{i \in T} \sum_{s \in S} (D_{i,s} - \overline{D})^2$, $SS_{res} = \sum_{i \in T} \sum_{s \in S} (D_{i,s} - \hat{D}_{i,s})^2$
then the R^2 is defined by :

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} = 1 - \frac{\sum_{i \in T} \sum_{s \in S} (D_{i,s} - \hat{D}_{i,s})^2}{\sum_{i \in T} \sum_{s \in S} (D_{i,s} - \overline{D})^2} \tag{4.1}$$

This score is between $-\infty$ and 1. A score of 1 means that all variance has been explained by the model. This score can be negative because a model can be arbitrarily bad. This score measures the explained ratio of variance. It is not a measure of the goodness of a model. A good model can have a low R^2 score, and a model that does not fit the data can have a high R^2 score.

4.3 Predicting Traffic Expectation

As presented in Section 4.1 the proposed model of the bike-sharing system is split into three parts : the prediction of traffic expectation ($\hat{\mu}_s(t,w)$), the prediction of the expected variance ($\hat{\sigma}_s(t,w)$) and the probability distribution of the model ($d(\mu,\sigma)$). In this section we model the traffic expectation ($\hat{\mu}_s(t,w)$). This part aims to predict, given the features selected in Section 3.6 (time and weather features), the traffic expectation. It corresponds to the 'mean estimation' box of Figure 4.1.

We remind that the expectation model has to predict two values per station using weather and time features. Two datasets are available : the first is the feature dataset, containing for each one of the 10 thousand hours of the study its weather and time features. The second one is the trip dataset, containing for each hour of the study (10 thousand) the number of departures and arrivals at each station ($2 \times n_{stations}$ numbers). For Montreal, this second dataset is a 10000×928 matrix. The naive approach is to build one model per station. This approach is not able to take into account of the similarities between stations. Another issue of the naive

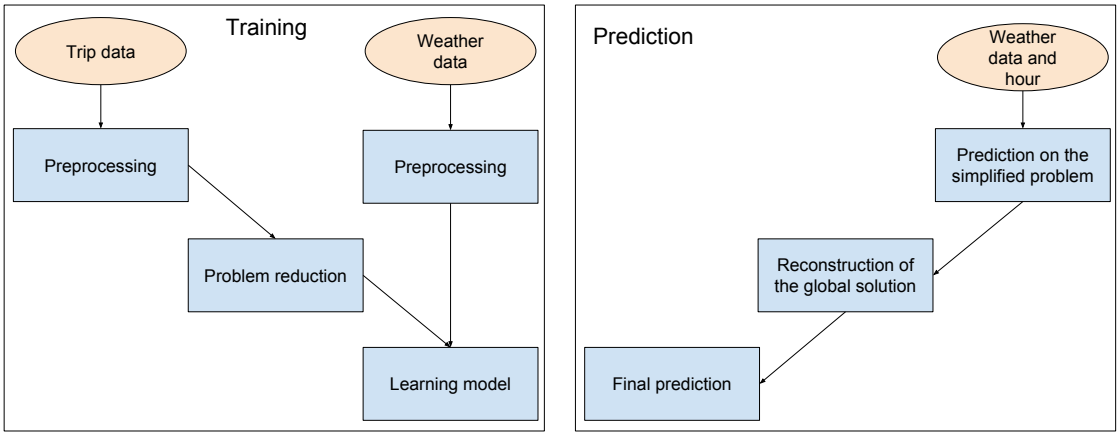

Figure 4.2 Model for mean prediction

approach is the stochasticity, the measured values are realizations of a random variable (X_s), and it is hard to find the expectation of this variable using only data from a specific station as we have to separate the influence of exogenous variable and randomness. These two effects may be correlated, especially with so many predictors to learn. Overfitting may become a real issue. To counter the stochasticity problem, we use neighboring stations. Then two problems remains : what is a neighboring station ? and how do we use this information ? We propose to solve these problems by considering neighboring stations in terms of user behavior, and by using this measure to reduce the problem to a smaller one. We transform the problem into a two-step one : first the reduction problem that simplifies the problem, then the prediction

problem that leans a predictive algorithm on the reduced problem.
Figure 4.2 presents how these parts interact. The reduction part decomposes the trip data into main behaviors, and keeps only relevant ones (reduction of the problem). Then a predictor learns to predict these extracted behaviors. These steps compose the learning phase. To recover a prediction on the whole network the process is reversed : a prediction is made on the main behaviors using the feature and the prediction part, then the reduction is reversed to recover a prediction over the whole network.

Formally the learning part learns a first a reduction method red and its pseudo-inverse inv_red to minimize :

$$loss(inv\_red(red(D)), D)$$

where $loss$ is scoring function to be defined and D the data matrix. Then a predictor $pred$ is learned to predict the reduced data :

$$pred(F) \simeq red(D),$$

where F is the feature matrix.

To get a prediction over the network we use the prediction function :

$$\hat{\mu}(t, w)_s = inv\_red(pred(F = (t, w)))_s,$$

where F is the features matrix of the data to be predicted, $pred$ is the same prediction function, and inv_red is the pseudo-inverse of red function. Since reduction methods project the data into a smaller space, they are not bijective and do not have an inverse. Thus, to inverse the reduction, a pseudo-inverse is built trying to limit as much as possible the information $loss$.

In the following sections we discuss how to choose $pred$ and red functions optimally.

To compare results of algorithms we slip the data into three sets, the training set, the validation set and the test set. The training set is used for model training. The validation set is used to refine models : hyperparameters selection, first comparison of algorithms, selection of best algorithms. The test set is kept for the last results, to confirm the results obtained on the validation set and to compare all algorithms. The test set is used in Section 4.8.

4.3.1 Literature

Predicting the traffic per station is a problem addressed in the literature, and several methods have been proposed. The Section 2.3 of the literature review focus on this part. We recall in this part the contribution the most related to our work. Chen et al. (2016) propose a cluster based approach. They reduce the problem using clusters, and restrict the problem to these clusters (no inversion). Clusters are computed using geography and station behavior. The aims of their problem is to predict over demand probability for each cluster. Rudloff and Lackner (2014) propose a model that does not reduce the problem. Some extra features are added indicating the status of neighboring stations. Li et al. (2015) propose a model based on a cluster reduction. Their model cluster the station using their position and behavior, then estimate the intra-cluster and inter-cluster traffic. Gast et al. (2015) propose a model similar to Rudloff and Lackner (2014) one, using an estimation of the demand per hour and station. However they do not specify how they estimate their parameters. Some articles do not specify how model parameters are estimated (Nair et al., 2013; Gast et al., 2015), then they can not be compared to our methods.

4.3.2 Reduction Methods

The reduction aims to simplify the prediction problem. The original prediction problem is to predict $2 * n_{stations}$ values, two per station. This part aims to reduce this number to a smaller one. The large number of objectives poses several problems. First of all, it requires a great computation capacity (time and power) and memory (RAM and ROM). This point may not be critical for this application as the learning part is not done online, but reducing the computation time is useful to build more complex models. Then, such a model would be ineffective since it would learn many times very similar models. In this part we will try to reduce the problem so as to limit the computing capacity (time and power) necessary, while maintaining similar performance.

To reduce the problem, we transform the trip count matrix (number of trips per hour and station) to a smaller matrix. The objective is to reduce as much as possible the number of columns of the trip matrix while maintaining most of the matrix information. This reduction is achieved by grouping similar stations together or by extracting main behaviors. Then we inverse this reduction by expressing station count data with the extracted behaviors. The prediction problem is transformed from predicting the traffic for each station each hour to predicting the extracted behaviors each hour. In practice we reduce the number of columns a factor of 100.

This reduction process should erase some noise, abnormal phenomenons and critical stations

bias (full or empty stations), by averaging behavior between stations. However, simplifying the problem also means to loose some information. A too severe reduction would be very imprecise as to much information is lost and a too light reduction will learn some noise, losing in efficiency and computation power. A compromise has to be found between precision (conserve most information) and problem complexity (reduce the number of columns).
To predict the traffic over the network, the reduction (grouping stations or extracting main behaviors) has to be reversed. However, as we reduce the problem, we lose some information, then the reduction cannot be reversed, then we define for each reduction method *red* a pseudo-inverse method *inv_red*. This subsection aims to present some reduction methods and the related pseudo inverse function. In this work we restricted ourselves on reduction methods based on the following algorithms :

— Identity
— Sum
— SVD
— Autoencoders
— Kmeans
— Gaussian Model

We will explore what are the advantages and disadvantages of each technique, and we will make a first selection of algorithms.

Most of presented reduction (resp. pseudo-inverse) methods are linear, except for the autoencoder. This means that there exists a matrix R (resp. R') such that $red(D) = D \cdot R$ (resp. $inv\_red(D') = D' \cdot R'$)

Note : in this subsection we focus only on the trip count matrix, features (time and weather) are not considered. The model uses the features only in the prediction part. This subsection extract the intersect behaviors of stations.

Identity

This method is a baseline, no reduction is performed, the data is kept as it is. This approach is the only one with no information loss. This method is very easy to perform (nothing is done) but leads to a more complex prediction model because of the number of dimensions.

The reduction equation is the following :

$$red(D) = D \tag{4.2}$$
$$inv\_red(D') = D' \tag{4.3}$$
$$R = R' = I \tag{4.4}$$

Where I is the identity matrix.

Sum

This method reduces all dimensions to one by summing them together. It is very simple but leads to a big loss of information. It is used as a naive benchmark to measure the improvement made by other algorithms. The equation representing the operation of this method are :

$$red(D) = \sum_{s \in S} D_{\_,s} = D \cdot R, \qquad \text{with } R = [1, ..., 1] \tag{4.5}$$
$$inv\_red(D') = R' \cdot D' \qquad \text{with } R' = \frac{1}{n_s}[\bar{D}_{\_,1}, ..., \bar{D}_{\_,n_s}] \tag{4.6}$$

The resulting dimension is the sum of all trips (arrivals + departures) on the network each hour. This method is equivalent to taking the mean number of trips. To inverse the reduction, the predicted number of trips is distributed into all stations, proportionally to the average number of trips per station.

SVD

The Singular Value Decomposition (SVD) is a very common reduction technique. It uses the following theorem to reduce the dimensionality of a matrix :
D is a $n_t \times n_s$ matrix in $\mathbb{R}^{n_t n_s}$ of rank r then it can be factorized as follows :

$$D = U \cdot \Sigma \cdot V^T$$

Where U and V are unitary orthogonal matrices of shape $n_t \times r$ and $r \times n_s$ and Σ is the $r \times r$ diagonal matrix of singular values, ordered from the biggest to the smallest. The SVD reduction consists in selecting the most significant dimensions (in terms of singular values), and to set to zero all the others. Let's note $V' = [V_{\_,1}, ..., V_{\_,k}]$, the first k columns of V, for a predefined value of k. Then the reduction method computes :

$$red(D) = D \cdot V' \tag{4.7}$$

and the inverse transformation :

$$inv\_red(D') = D' \cdot V'^{T} \tag{4.8}$$

The orthogonality of V justifies this pseudo-inverse. This method is very close to the PCA method, which centers and norms the data before computing this transformation. In our problem normalization is not essential as all dimensions represent trip counts.
The *scipy* library is used to compute the SVD.

We apply the SVD reduction to the trip matrix (number of trips per station and hour), to recover a much smaller matrix, explaining most trip variance. Stations are replaced with some behaviors. Stations trip counts can be recovered using a linear combination of the extracted behaviors.

Autoencoders

Autoencoders are artificial neural networks that try to learn a representation of a dataset, usually for dimensionality reduction. It is an unsupervised learning algorithm (Hinton and Salakhutdinov, 2006). The autoencoder's neural network tries to predict x given x. The trick is that to predict x the network has to pass the information to a lower-dimensional space. Then it learns a representation of x in a smaller space. This smaller layer cuts the network into two parts, the encoding part and the decoding part. The first part creates a representation in a lower dimensional space of the input ($red(D)$ function), the second reconstructs the input from its representation in the lower dimensional space ($inv\_red(D)$ function). The two parts are trained together, and separated to encode or decode some input vector. The autoencoder is a non-linear algorithm. The simplest autoencoder should perform nearly a PCA on the data (same subspace) (Baldi and Hornik, 1989). Therefore a more complex one should be more effective than the usual PCA or SVD. The non-linearity of the autoencoder is essential to achieve such performance. The drawback of this improvement is that this non-linearity may lead to inconsistent results, where small differences in the reduced dimensions lead to big ones in the complete space. Regularizations are essential to build a robust autoencoder. Noise is added to the data to improve the autoencoder robustness. This new autoencoder tries to predict x given a noisy version of x. The autoencoder computes :

$$red(D) = f_n(f_{n-1}(...f_1(D \cdot W_1 + b_1)...) \cdot W_n + b_n) \tag{4.9}$$

$$inv\_red(D) = f_{2n}(f_{2n-1}(...f_{n+1}(D \cdot W_{n+1} + b_{n+1})...) \cdot W_{2n} + b_{2n}), \tag{4.10}$$

Where $f_i, i \in 1,..,2n$ are activation functions, $W_i, i \in 1,..,2n$ some weight matrices and $b_i, i \in 1..2n$ constant vectors.

Clustering

All clustering technique works similarly in terms of reduction operations and pseudo-inverse. They differ only on the computation of clusters. Each clustering techniques minimize a particular function.
Clusters can be computed in many ways. In this thesis we cluster stations based on the observed demand for each hour of the training dataset. We ignore any other characteristic of stations (location, environment, ...). This choice agrees with the information loss minimization objective.
The reduction of the data is computed by averaging the stations demand through each cluster. Therefore each dimension corresponds to a cluster's centroid. More formally :

$$R_{i,j} = \begin{cases} \frac{1}{|K_j|} & \text{if } i \in K_j \\ 0 & \text{else} \end{cases} \quad \forall i \in S, j \in \{1,...k\} \tag{4.11}$$

where k is the number of clusters and K_j is the j-th cluster. Then

$$red(D) = D \cdot R \tag{4.12}$$

To invert the reduction, R' is computed as :

$$R'_{j,\_} = \frac{1}{\sum_{i \in K_j} \bar{D}_{\_,i}} [\bar{D}_{\_,1} \times \mathbf{1}_{1 \in K_j}, ...\bar{D}_{\_,n} \times \mathbf{1}_{n \in K_j}] \quad \forall j \in \{1,..,k\} \tag{4.13}$$

and the pseudo-inverse operation is defined as :

$$inv\_red(D') = D' \cdot R' \tag{4.14}$$

Kmeans

Kmeans is a heuristic method that aims to solve the problem of minimizing the squared distance of points to the cluster centroid. Aloise et al. (2009) show that this problem is NP complete (finding the minimum). Then kmeans is a heuristic that aims to minimize :

$$\sum_{s \in S} (D_{\_,s} - c_{K_s})^2, \tag{4.15}$$

where K_s is the cluster of station s and c_K is the centroid of cluster K. This methods proceed by assigning points to the closest centroid, then recomputes the centroid. These two steps are repeated until convergence. Kmeans is run ten times to compute the clusters, then the reduction an inverse reduction methods are computed using the previous paragraph methods. It is lunched ten times to achieve better results.

Gaussian Mixture

Gaussian mixture is also a clustering algorithm, that interpret the data points as realizations of Gaussian variables. This algorithm selects k Gaussian variables and assign each point to the Gaussian variable that maximizes the likelihood of the point. The parameters of the random Gaussian variables are determined using a Expectation Maximization algorithm that maximizes the likelihood of the data points. This algorithms repeats until convergence the two steps :

— assign points to a cluster

— compute the optimal parameters of each cluster (maximization of the likelihood)

The algorithm is stochastic, then to achieve better performance, it is lunched ten times and the best result is kept. The kmeans method is a special case of Gaussian Mixture where the variance parameters of the random variables are equals.

4.3.3 Reconstruction loss and reduction optimization

Reduction methods proposed in the previous section can be compared in terms of reconstruction losses (RL) defined as :

$$RL = mean_{(s,t)\in S\times T}\Big(D_{s,t} - inv\_red(red(D))_{s,t}\Big)^2 \tag{4.16}$$

This reconstruction loss quantifies the information lost from the problem simplification. To interpret these values this score is normalized :

$$RL_{norm} = 1 - \frac{mean_{(s,t)\in S\times T}\Big(D_{s,t} - inv\_red(red(D))_{s,t}\Big)^2}{mean_{(s,t)\in S\times T}\Big(D_{s,t} - \bar{D}\Big)^2} \tag{4.17}$$

This score is a R^2 score adapted to the reduction methods. It indicates the proportion of data kept in the reduction process. These scores are presented in table 4.1. The following paragraph aims to select for each reduction method the best hyperparameters

SVD

An analysis of the information loss is performed for the SVD. Figure 4.3 presents the ratio of variance explained by each principal component (discrete derivative of RL). In this part we want to minimize the dimension of the reduced problem while minimizing the reduction loss. However the reduction loss only gives a good insight on the quality of the reduction. A very high information loss will give a bad prediction model. However, a very low information loss will not necessarily give a better model than medium one because of the stochasticity of the demand (Section 4.4 gives numerical results). Figure 4.3 shows that a five dimensions space explains most of the data with a very low number of dimensions. This result will be refined later in the work (Section 4.4) optimizing the model precision. The second Figure represents the first components of the SVD, grouped by hours of the week (there are 168 hours in one week). The resulting dimensions could be quite difficult to interpret, they represent the

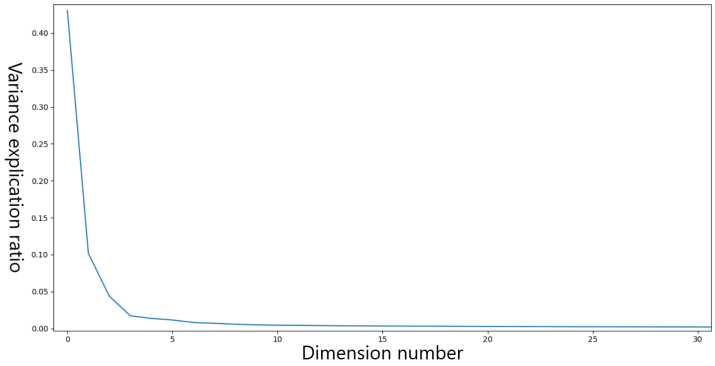

Figure 4.3 Explained variance ratio per singular value

main patterns in the data. The first component represents the mean number of trips per hour, the second is used to balance stations between morning activity and evening activity. The third makes possible to dimension peaks correctly. The following components are noisier and harder to interpret.

Autoencoder

A grid search has been performed on the hyperparameters of the Autoencoder to find the best ones. The grid search has been performed on :

— the number of hidden layers (between 0 and 5)

— the size of layers (between 5 and 1000)

— the dropout value (between 0 and 1)

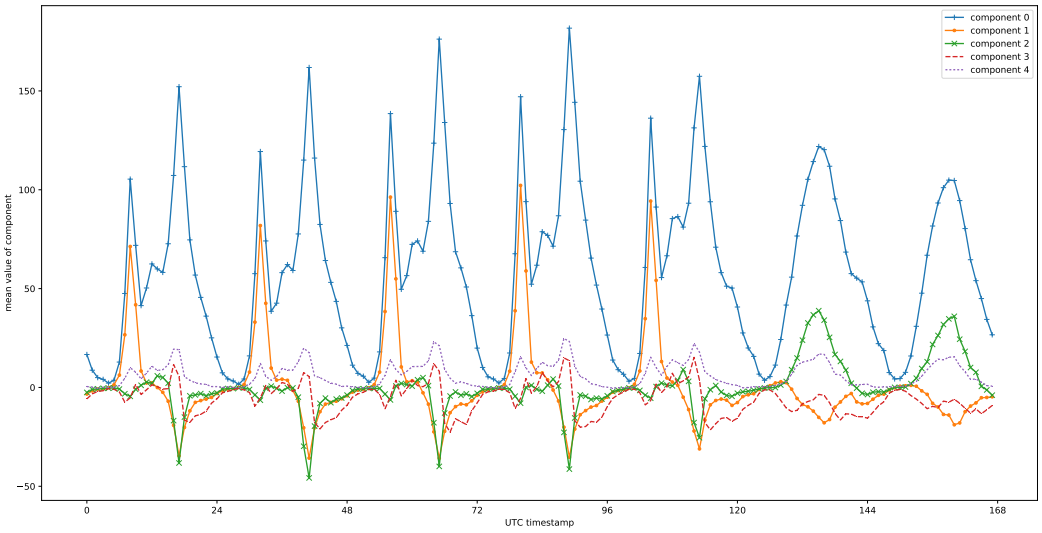

Figure 4.4 SVD first components

— the batch normalization (yes/no)

— the activation functions ('relu', 'tanh', 'sigmoid')

— the learning rate

The final solution had 2 hidden layers with 70 and 35 neurons. The dropout is not used. Batch normalization gave good results. The activation function is a ReLU (Rectified Linear Unit). A learning rate of two is used, with an adadelta optimizer. Some noise is added to the input data and on the encoded data (middle layer).

Clustering

The data points for the clustering methods had ten thousand dimensions (one per hour of the study), with about a thousand points. Thus to counter the curse of dimensionality, hours were aggregated per week. Each new coordinates now represent the average number of trips done in this particular station and in that hour of the week. The new data had 168 rows.

Kmeans

We computed the RL score for a number of clusters between 1 and 50. The evolution of this score as a function of the number of clusters is displayed in Figure 4.5. This error measure

decreases with the number of clusters as expected. However the improvement of each new cluster is low after 10 clusters. As this error tends to zero, (reached for n_s clusters) we have to choose an optimal number of clusters. Figure 4.5 shows that after ten clusters the error decreases much more slowly, hence we select ten clusters, and ignore the remaining information. This result will be refined later in this work (Section 4.4) by considering the precision scores. For now, the only available measure is the reconstruction loss, which is not a perfect measure of the performance of the reduction because it does not consider the denoising and stochasticity.

The second figure of 4.5 shows the average behavior of each cluster centroid. These centroids show main station behaviors of the network. We can see morning stations (cluster 6), evening stations (cluster 1), balanced stations (cluster 5), leisure stations (cluster 7) and small stations (clusters 2,3,4).

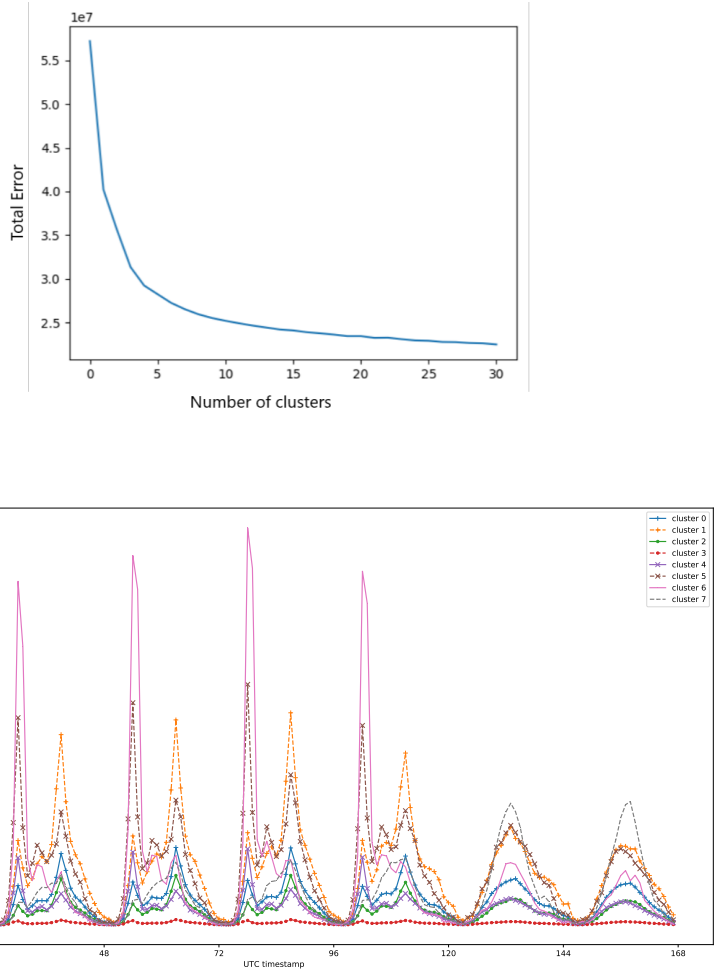

Figure 4.5 Kmeans analysis

Gaussian Model

The Gaussian Model algorithm is very close to the kmeans one. So, the same analysis has been made and also ten clusters were selected. Results were very similar. The *scipy* library is used to implement it.

Summary

All these methods try to reduce the complexity of the problem. Table 4.1 summarizes the properties and results of each method. It presents the number of dimensions of the reduced problem the RL score and the RL_{norm} score.

Proposed methods explain between 30 and 72% of the data with 2 to 10 dimensions. The best algorithm in term of RL_{norm} is the autoencoder that needs only 5 dimensions to explain 70% of the data. The best linear method is the SVD, which manages to conserve 68% of the data, with only 5 dimensions and 71% with 10 dimensions. The clustering methods are, by construction, worse than dimensionality reduction techniques as SVD, because they are special cases of it. However, they manage to conserve 63% of the data with 10 dimensions. This reduction operation has a cost : the reduced data is close to the original one, but some information is lost from approximations. However, this information may be only noise and, as we try to predict the expected number of trips, this loss may improve the performance. Consequently, the data is smoothed during the reduction process. This table shows that each

Table 4.1 Summary of principal characteristics of reduction methods

method	dimension	RL	RL_{norm}	training time (s)
Identity	2*n_stations	0	1	0
sum	1	6.36	0.20	0
SVD	5	2.58	0.68	0.46
SVD	10	2.34	0.71	0.41
autoencoder	5	2.45	0.70	124.03
autoencoder	10	2.28	0.72	130.31
kmeans	5	3.08	0.62	8.59
kmeans	10	2.97	0.63	9.49
Gaussion Model	5	3.16	0.61	0.30
Gaussion Model	10	2.96	0.63	0.44

method is able to conserve most of the information except the sum reduction. Clustering methods are able to conserve most of the data, however, the SVD and the autoencoder are able to conserve more information. The best algorithm is the autoencoder even if the SVD

is very close and easier to compute : it takes 280 times less time to compute a SVD than to learn the autoencoder. Its non-linearity makes it also harder to predict. All methods loose at least 25% of the data information. We do not know yet what is this information loss. However we suspect that it is mainly noise, which may not impact the result of the model.

4.3.4 Prediction Methods

Once the problem has been reduced, the next step is to model and predict the extracted behaviors. This step is the central one on traffic forecasting, as it determines the goodness of the proposed model.

The previous subsection focus on defining the *red* function of the learning equation (Section 4.3) :

$$pred(F) \simeq red(D)$$

This subsection aims to define and learn the *pred* function. We use several learning algorithms to minimize the difference between $pred(F)$ and $red(D)$. We examine this difference in terms of $RMSE$, $RMSLE$, MAE, $MAPE$ and R^2. We propose to learn the *pred* function using an approach based on the following algorithms :

— Mean predictor

— Linear Regression and regularization's

— Neural Network (Multi Layer Perceptron)

— Decision Tree

— Random Forest

— Gradient Boosted Tree

The final predictor is composed by the *pred* function learned is this subsection and the inverse reduction method defined in the previous subsection :

$$\hat{\mu}_s(F) = inv\_red(pred(F))_s$$

This subsection focus on using current features to predict the traffic during current hour. More complex models can be built using data from previous time steps, some of these approaches are tested in Section 4.5.1

Notation

We complete the notation of Section 4.2 by adding the following ones :

U is the set of reduced dimensions. For the identity reduction it corresponds to S.

n_U the dimension of the reduced space $n_U = |U|$

Y is the reduced data matrix, the output of the reduction method. It is a $|T| * n_r$ matrix

$\hat{Y}_{algo}$ is the estimation of Y using $algo$ for the prediction of Y.

Mean predictor

This model is a naive baseline. This predictor predicts a constant number for each dimension equal to its mean. This model does not consider the features matrix F, it is based only on the reduced trip matrix Y. It's the most naive approach. It is a baseline used to compare prediction algorithms, that must score better than this very naive method. Mathematically this method does :

$$\hat{Y}_{Mean}(F) = \begin{bmatrix} \bar{Y}_{\_,1} & ... & \bar{Y}_{\_,n_R} \\ : & : & : \\ \bar{Y}_{\_,1} & ... & \bar{Y}_{\_,n_R} \end{bmatrix} \tag{4.18}$$

Linear Regression

This model is the most used model in the literature, because of its simplicity and performance. This model tries to express the data as a linear combination of the features. It minimizes the sum of squared errors between predicted and true values. It also gives a good insight on the significance of each feature. This method is linear, then, to be able to learn non-linear relationships, some features are transformed into categories (Rudloff and Lackner, 2014; Li et al., 2015; Borgnat et al., 2011). This transformation is done for the hour and week-day features. Section 4.4 shows the improvement of this transformation. Formally this method compute :

$$\hat{Y}_{lin}(F) = [F|1] \cdot W \tag{4.19}$$

Where W is a $(n_F + 1) * n_R$ weight matrix. This parameter is determined by minimizing $\sum_{t \in T} (Y_{t,\_} - \hat{Y}_{lin}(F_{t,\_}))^2$.

Regularized Linear Regressions

Some regularized form of linear regression has been explored. The most common are the Lasso regression and the Ridge regression. The lasso regression minimizes $\sum_{t \in T} (Y_{t,\_} - \hat{Y}_{lin}(F_{t,\_}))^2 + \lambda \sum_{w_{i,j} \in W} |w_{i,j}|$ where λ is the regularization coefficient. The Ridge regression minimizes $\sum_{t \in T} (Y_{t,\_} - \hat{Y}_{lin}(F_{t,\_}))^2 + \lambda \sum_{w_{i,j} \in W} (w_{i,j})^2$ with the same notations. These regularizations force the coefficients of the regression to be controlled (not too big). The formula to obtain

the prediction is the same as for a linear regression. These methods are analyzed in Section 4.4.

Neural Network

Neural Networks are a general learning model, composed of neurons. Each neuron receives some entries (In), multiplies them by a weight (W and b), applies an activation function (f) and returns the result. This result is then sent to other neurons. The operation made by one neuron is :

$$Out = f(W \cdot In + b)$$

A network of these neurons is built by branching the output of some neurons to the input of other neurons, and the input data to the input neurons. The final computation is :

$$NN(M) = f(W_{l_n} \cdot ...f(W_{l_2} \cdot f(W_{l_1} \cdot M + b_{l_1}) + b_{l_2})... + b_{l_n})$$

The learning part of this method consists into the adjustment of the weights. Neurons are usually split into consecutive layers. The difficulty of this method is to find an efficient network and to counter overfitting. The large number of hyperparameters makes the conception of such network very difficult. Common techniques to improve and stabilize the performance of the network are dropouts, skip connections, data augmentation, and more advance learning techniques as Adam or Nadam as well as regularization (L1 or L2). (Kingma and Ba, 2014; Dozat, 2016; Srivastava et al., 2014; He et al., 2016)

Decision Tree Algorithms

Several decision tree algorithms have been tested on the data, from the simplest one (decision tree) to more complex ones as random forest and gradient boosted trees. These algorithms use decisions rules (tree structure) to predict the expected traffic. Random forest uses several trees and combines their result to make a prediction. Gradient boosted tree uses several trees, learned on the residual of the previous tree to predict the traffic. These models are widely used in the bike-sharing literature (Yin et al., 2012). The decision tree determines the next decision point maximizing the entropy gain Fayyad and Irani (1992). It chooses the decision value that better separates the data (χ^2). Yin et al. (2012) shows that the gradient boosted tree outperforms the linear regression, the random forest and the Support vector regression to predict the global traffic on the network with time and weather features.

4.4 Expectation Hyperparameters Optimization

After defining the set of algorithms that we test to build our model, we optimize their hyperparameters to get better results. This section and the following subsections aim to optimize the hyperparameters of all proposed models. However the large number of possible models make this optimization very long, therefore we choose to approximate this optimization. This approximation is based on the fact that reduction methods extract very similar behaviors from the data. Therefore using different reduction methods do not change the learning problem. The prediction algorithm has to learn very similar behaviors. Hence the optimal hyperparameters of prediction algorithm stay unchanged if we change reduction method. To investigate whether hyperparameters have similar effect we compare the $RMSE$ of 3000 models trained on different set of parameters. These model were composed of a random forest predictor and one reduction method between kmeans, SVD, GM and autoencoder. We display in Table 4.2 the correlation between these $RMSE$ scores. The table shows correlations higher than 99%, therefore hyperparameters can be optimized using only one reduction method. The best set of parameters found will be quasi-optimal for other reduction methods. Therefore, we chose to optimize the hyperparameters only for the SVD reduction and so this value is assumed to be quasi-optimal for other reduction methods.

Table 4.2 correlations between $RMSE$ score of four different algorithms, varying the random forest hyperparameters.

	kmeans	SVD	GM	autoencoder
kmeans	1	0.9983	0.9999	0.9903
SVD	0.9983	1	0.9983	0.9921
GM	0.9999	0.9983	1	0.9903
autoencoder	0.9903	0.9921	0.9903	1

4.4.1 Selection of the number of dimension, optimization of reduction methods

In the reduction part we proposed several methods to reduce the problem and proposed a dimension value for the reduced spaces. In this section we refine this measure to get the best algorithm, while keeping a small output space. We preselected in Section 4.3.2 the dimension of the reduced space based on the information loss of reduction methods. However, this loss does not represent our final objective, which is to predict future data. Then we can examine the evolution of the precision with respect to the reduction dimension. Figure 4.6 displays the $RMSE$ for kmeans and SVD reductions. This figure shows that a good choice of dimension for the SVD and kmeans clustering is 10. This value by minimizing the $RMSE$ for the SVD.

For the kmeans, the precision can be slightly improved by adding more clusters, however, as we also want to minimize the cluster number, ten clusters offer a good balance between precision and complexity. These reduction dimensions are kept for the rest of the work.

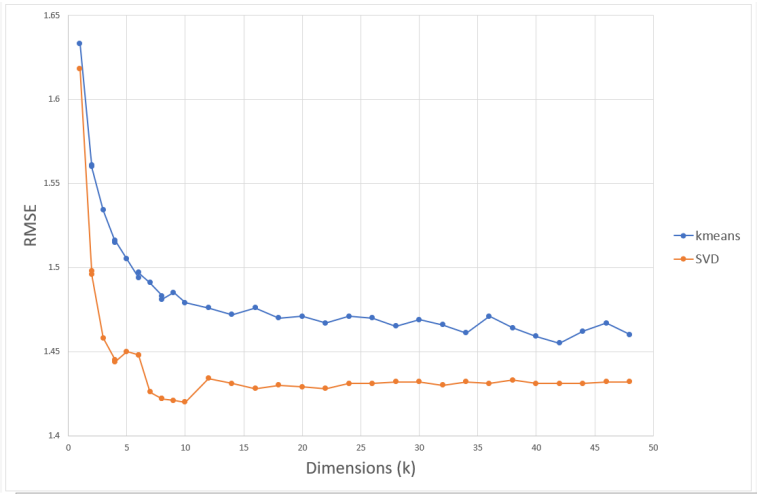

Figure 4.6 $RMSE$ per dimension for kmeans and SVD

4.4.2 Linear Regression

The linear regression has no hyperparameters, the only tuning possible is on the features. In Section 3.6 we built categorical continuous versions of some features. For linear methods, categorical features are more suited as they allow to learn a non-linear function. The following table compares the different methods, the last line corresponds to the model using a linear regression predictor and no reduction (identity). We compare its precision to a model using a linear regression predictor and a SVD reduction. This table confirms our expectations,

Reduction	features	$RMSLE$	$RMSE$	MAE	$MAPE$	R^2
SVD	continuous	0.602	2.438	1.326	0.612	0.432
SVD	categorical	0.542	2.104	1.16	0.525	0.577
identity	cat	0.541	2.097	1.155	0.523	0.58

categorical variables give better results. The linear regression performs as well on the SVD reduction as on the identity reduction (no reduction). This shows that no valuable information is lost during this reduction step and that reducing the problem is pertinent. As we kept some highly correlated variables, to limit weight values and overfitting, we also used ridge and lasso regressions. A grid search has been performed on the Lasso and Ridge regression (L1 and L2 regularization) to find the best λ (importance of regularization). Values between 10^{-10}

and 10^{20} were tested. The best values found were 2.55 (Ridge) and 10 (Lasso). However, the regularization is not able to significantly improve the results (<1% of improvement). The linear regression performing as well as the regularized one, and thus only the linear model without regularization is considered in the experiments.

4.4.3 Decision Tree

Decision trees capture non-linear relations between features and outcome variables, therefore using categorical features is not essential. The two options are tested, and results are visible in the following table. It appears that continuous features give better results for this model. This conclusion is true for all decision tree based algorithms, and then used for all models. The last line shows that using a SVD reduction gives slightly better results than using no reduction (identity).

reduction	features	$RMSLE$	$RMSE$	MAE	$MAPE$	R^2
SVD	continuous	0.499	1.99	1.06	0.446	0.622
SVD	categorical	0.522	2.033	1.107	0.476	0.605
identity	cat	0.527	2.057	1.11	0.466	0.596

A grid search has been performed on the decision tree to find the best parameters for the tree. The algorithm is optimized on the minimum number of samples per leaf (20 to 1000) and the maximum depth of the tree (1 to 20). The minimum number of samples per split is also studied. However setting a minimum number of samples per leaf works better. We found that the best tree has a **max depth of 11** and a minimum of **20 samples per leaf**.

4.4.4 Random Forest

A similar optimization is performed on the Random Forest. We optimize the random forest using between 2 and 1000 estimators, a max depth between 2 and 20, and a minimum number of samples per leaf between 1 and 10000. The best forest found have **92 trees**, with a **max depth of 14** and a minimum of **3 samples per leaf**. The following table shows the performance of the random forest with a SVD reduction. The improvement is significant compared to the decision tree predictor.

features	$RMSLE$	$RMSE$	MAE	$MAPE$	R^2
continuous	0.491	1.923	1.037	0.442	0.647
categorical	0.512	1.956	1.085	0.475	0.634

4.4.5 Gradient Boosted Tree

As for the previous algorithms, a grid search is performed on the hyperparameters of the gradient boosted tree. We search for a number of estimators between 5 and 1000, a max depth between 1 and 20 and a learning rate between 10e-5 and 1. We found the best results for **150 estimators**, a **learning rate of 0.1** and a **max depth of 5**.

features	$RMSLE$	$RMSE$	MAE	$MAPE$	R^2
continuous	0.487	1.882	1.023	0.427	0.662
categorical	0.494	1.907	1.041	0.437	0.652

4.4.6 Multi-Layer Perceptron (MLP)

The neural network used for this problem is quite small because the number of inputs does not justify a bigger network and big networks are harder to train and optimize. Several networks have been tested varying the following hyperparameters :

— number of layers (between 1 and 10)

— size of layers (between 10 and 1000)

— the dropout

— the regularization (parameter and L1 or L2)

— skip connections

— batch normalization

— activation functions (relu, sigmoid, tanh)

— learning rate

The adopted solution is a multi-layer perceptron, with 3 hidden layers of 60, 30 and 15 neurons, a tanh activation function, a dropout of 0.6, with batch normalization after each layer, and a learning rate of 0.01. The Nadam (Dozat, 2016) optimizer is used. The performance of the MLP applied to the SVD reduction is

features	$RMSLE$	$RMSE$	MAE	$MAPE$	R^2
continuous	0.603	2.446	13.29	0.613	0.428
categorical	0.545	2.124	1.17	0.531	0.569

4.5 Test of mean traffic estimation model

Section 4.3.3 already presented some results on reduction methods. It evaluates reduction methods by their capacity of conserving information, while reducing the problem complexity. This section aims to compare all algorithms, by comparing the precision of each method. As for previous tests, algorithms are learned on the training set, and tested on the validation dataset. The test data set is kept for the last results.
We built several algorithms to predict the expected number of trips per station (reduction + prediction method). This section compares the precision of each method and their computing time (learning and prediction time). All algorithms in this section are learned on the training set (64% of the data) and tested on the validation set (16% of the data). Final results will be presented in Section 4.8

4.5.1 Test of Data Transformations

Before comparing all algorithms, we test some transformation on the data to see how effective they are. In this section, we are going to focus on some data transformation and their impact on learning. First, we test a log transformation of Y the trip data (Wang, 2016), then we try to decorrelate the features F. Next we try to normalize the trip data D. Finally, we tried to increase the features with features from previous hours. To save space and readability we displayed only the results of the best working models. The conclusions of this section also apply to other models.

Log Preprocessing

Wang (2016) predicted in his article the log of the global traffic on the network. This subsection tests this approach and concludes about its effectiveness in the traffic prediction per station. This transformation makes algorithms minimize the $RMSLE$ instead of the $RMSE$.

Therefore, we replaced the objectives (Y) by their log ($Y' = log(Y)$). Then we trained the same algorithms, and computed the new errors. We displayed the best results found in table 4.3. It appears that this transformation decreases the precision of the predictor, therefore this transformation is not used for our model.

Table 4.3 Precision of the log transformation compared to the normal data

algorithm	log	$RMSLE$	$RMSE$	MAE	$MAPE$	R^2
SVD-GBT	No	0.487	1.882	1.022	0.428	0.661
SVD-randomForest	No	0.490	1.917	1.036	0.441	0.649
kmeans-GBT	No	0.506	2.003	1.087	0.465	0.617
kmeans-randomForest	No	0.507	2.018	1.092	0.469	0.611
SVD-GBT	Yes	0.683	2.068	1.450	0.883	0.591
SVD-randomForest	Yes	0.690	2.095	1.468	0.889	0.581
kmeans-GBT	Yes	0.702	6.448	1.658	0.925	-2.9
kmeans-randomForest	Yes	0.709	9.593	1.726	0.943	-7.79

Feature Decorrelation

As some features are highly correlated, the idea is to decorrelate them using a PCA. This subsection tests the influence of this decorrelation on the algorithms. This way resulting features are independent and represents orthogonal components of the features. The model architecture is kept the same but features are replaced by their decorrelated versions. As for the log transformation, some models are learned on these features and results are displayed in table 4.4. This table shows that this method makes the prediction harder. It is then not used in the rest of the article.

Table 4.4 Impact of the decorrelation on the precision

algorithm	decorelation	$RMSLE$	$RMSE$	MAE	$MAPE$	R^2
SVD-GBT	No	0.487	1.882	1.022	0.428	0.661
SVD-randomForest	No	0.490	1.917	1.036	0.441	0.649
kmeans-GBT	No	0.506	2.003	1.087	0.465	0.617
kmeans-randomForest	No	0.507	2.018	1.092	0.469	0.611
SVD-GBT	Yes	0.901	2.327	1.872	0.492	-0.77
SVD-randomForest	Yes	0.954	2.717	2.049	0.506	-2.16
kmeans-GBT	Yes	0.987	2.835	2.072	0.550	-15.26
kmeans-randomForest	Yes	0.990	2.966	2.086	0.570	-23.65

Normalization of stations

In this subsection we evaluate the effectiveness of normalizing the station behavior. The data matrix is replaced by :

$$D'_s = D_s / \sum_{h \in hour} D_{s,h}$$

This transformation aims to value big stations and small ones equally. The normalization is already used for clustering techniques, as it makes more sense to compare station behavior more than the actual traffic size. We compared the errors to the one made without normalizing the data. Table 4.5 presents these results. No difference is visible in the performance of the algorithms for the SVD.

Table 4.5 Impact of the normalization on the precision

algorithm	normalization	$RMSLE$	$RMSE$	MAE	$MAPE$	R^2
SVD-GBT	No	0.487	1.882	1.022	0.428	0.661
SVD-randomForest	No	0.490	1.917	1.036	0.441	0.649
kmeans-GBT	No	0.506	2.003	1.087	0.465	0.617
kmeans-randomForest	No	0.507	2.018	1.092	0.469	0.611
SVD-GBT	Yes	0.487	1.873	1.020	0.429	0.665
SVD-randomForest	Yes	0.496	1.924	1.045	0.454	0.646
kmeans-GBT	Yes	0.506	2.003	1.087	0.465	0.617
kmeans-randomForest	Yes	0.507	2.018	1.092	0.469	0.611

Using previous hour features

As for the global prediction, we tried to use features from previous hours (weather) to improve the prediction of the current one. We tried with several sets of hours, using up to 72 hours before the current one. Figure 4.7 displays the evolution of scores with the set of hours. The first set ([]) represent the precision with only the current hour features. While numbers in brackets represent the considered hour features. This figure shows that using previous hour features has no real impact on the prediction. The variation of scores is less than 2% of the scores.

Using previous hour real traffic, improving the model using time series

This section is similar to the previous one as it uses data from previous hours to refine the prediction. In this subsection we transformed our model to a time series model : we used the traffic of previous hours and the error made by the model during previous hours as features of current hour prediction. Previous models used only known (time) and predicted (weather) features, and could be applied regardless of the time horizon. Predicting traffic using real traffic, impose to build specific models for each possible time horizon : predicting the traffic one hour ahead is not the same problem as predicting it 2, 3 or 6 hours ahead. To simplify the problem and avoid overfitting, we used as features the simplified traffic, i.e. the real traffic

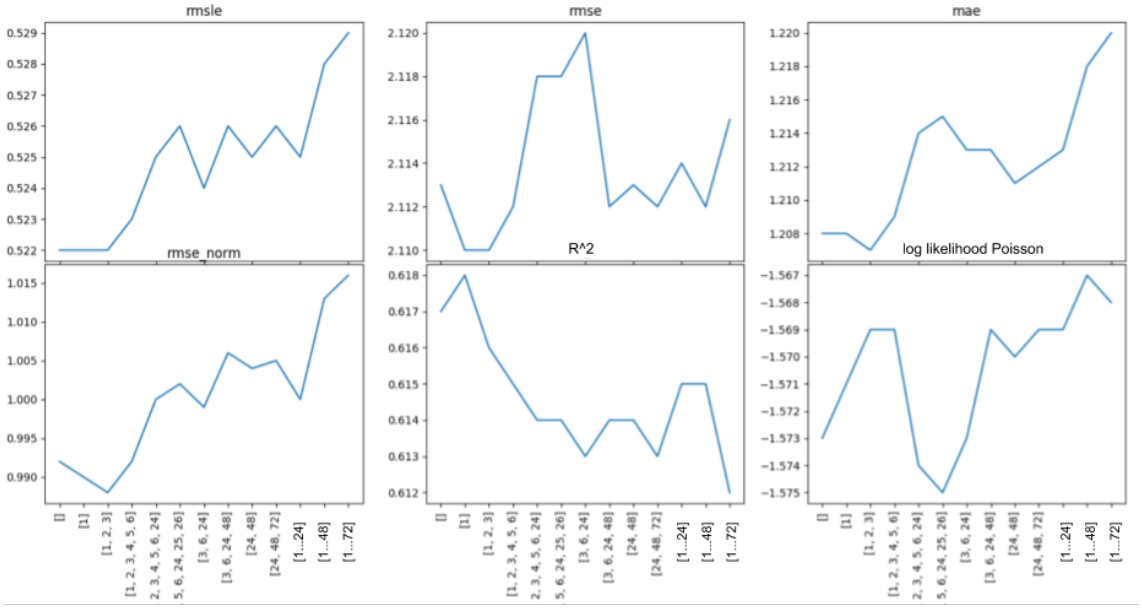

Figure 4.7 Scores evolution per time window

for which reduction was applied. Thus, we have 10 new features per hour instead of 1062 ($2 \times n_s$). The model built in the previous sections is called here the base model.

This section proposes three time dependent models. The first one is a non-linear autoregressive exogenous model (NARX) and the second one is a non-linear moving average exogenous model (NMAX). The non-linearity is introduced by the gradient boosted tree predictor. The third model is a variant of a NMAX model. These models are refinements of the autoregressive (AR) and moving average (MA) models introduced by Yule et al. (1927); Slutzky (1937). The NMAX is a general method. Leontaritis and Billings (1985) first proposed this model. The NMAX the non-linear version of the moving average exogenous model. This model is not widely used in the literature. This time series approach lead to similar results in the literature (Yoon et al., 2012; Li et al., 2015) and is out of the scope of this work. This subsection gives an insight on the improvement possible using such models.

First Approach The first approach is the simplest. It adds only the reduced traffic for the six previous hours to the features of the model. This model is represented in Figure 4.8.

Second approach The second approach replaces the observed reduced traffic by the error made by the base model. Then it learns a second predictor to predict the real traffic using features from the present hour (time and weather) and previous hours (errors made by the base model). This model is presented in Figure 4.9.

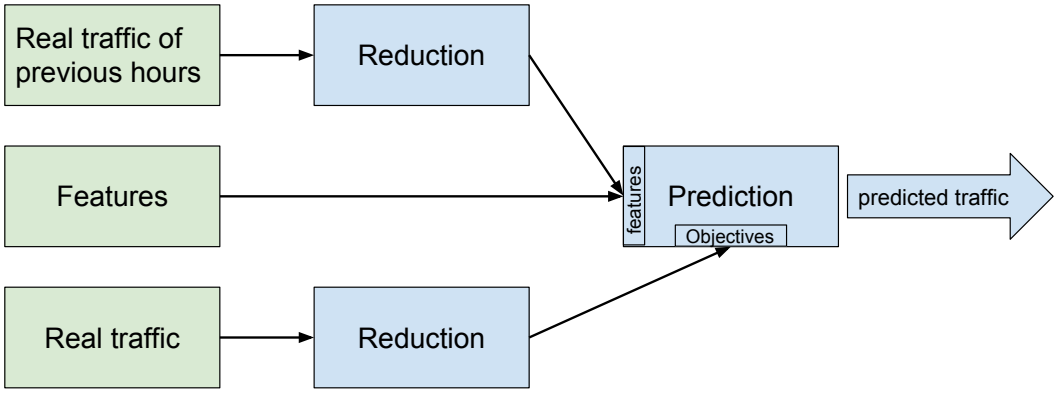

Figure 4.8 First time dependent model

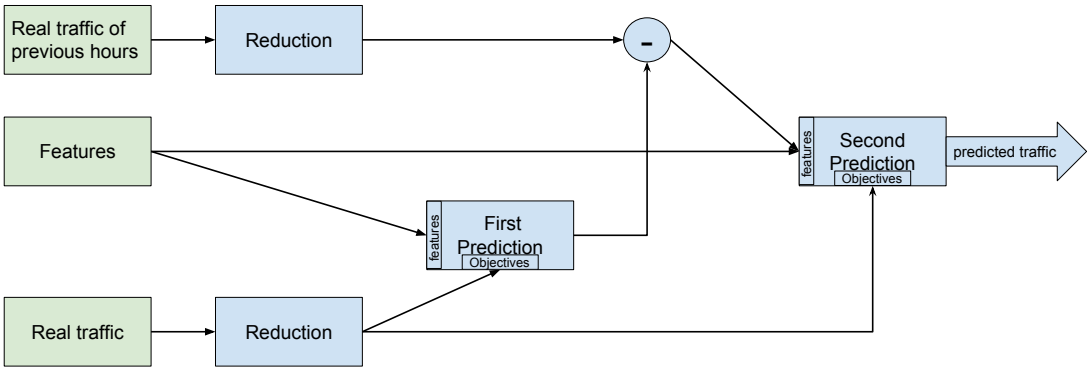

Figure 4.9 Second time dependent model

Results Table 4.6 presents the results of these three algorithms on two different tests set. The first one corresponds to the Jazz week, from the 29^{th} of June 2016 to the 9^{th} of July 2016. The second one is the validation set. This table shows that on the validation set, models 1 and 2 are equivalent. On the Jazz week, Model 1 performs better than model 2. However the difference in performance between the four models is very small, an improvement of 2-3% in terms of R^2 is achieved. Figure 4.10 shows the R^2 score of the two models during the Jazz week and during a common week of September. The horizontal axis is indexed by the number of days since the beginning of the test and the hour of each day. The R^2 score is displayed only between 2 p.m. and 3 a.m. to focus on hours where the model lacks of precision. the discontinuities of the lines show where hours were omitted. These graphs show that the Jazz week is much more chaotic than the September week. The difference between models is barely visible, but still present. We can see that the first model performs better in the tested week.

Table 4.6 average performance of time augmented models

	Jazz week		Validation set	
model	R^2	$RMSE$	R^2	$RMSE$
base model	0.533	1.578	0.624	1.682
first approach	**0.564**	**1.558**	**0.649**	**1.644**
second approach	0.554	1.565	0.645	1.649

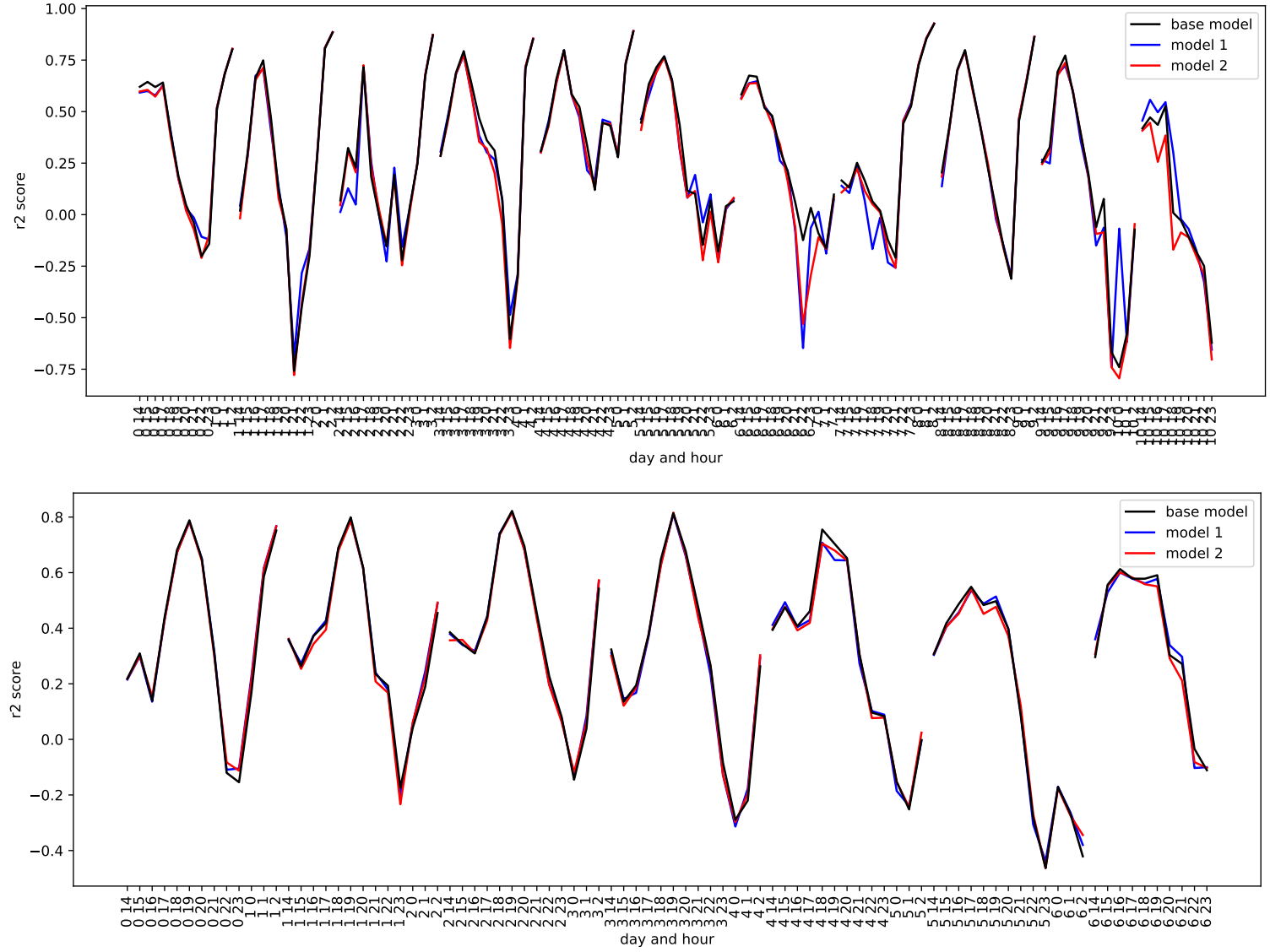

Figure 4.10 R2 score during the jazz week (first figure) and a September week (second figure), for hours between 2h p.m. and 2h a.m.

4.5.2 Features importance analysis

Decision tree based algorithms offer a simple way to compare the importance of learning features, implemented in the scipy library. The entropy gain associated with every feature is computed and aggregated to get the influence of every feature. We displayed in Figure 4.11 the importance of each feature for the decision tree, the random forest and the first three dimensions of the gradient boosted tree. A SVD reduction is performed before the learning of the prediction algorithm (decision tree, random forest or gradient boosted tree). Figure 4.11 shows that the most important features are the hour and the week day. The year, the visibility and the temperature are also significant. The gradient boosted tree, which performs better, is able to give also importance to the other features (humidity, day, month, ...). The gradient boosted tree also shows that the hour feature is very important (35 to 60%) to predict the traffic, but its importance decreases with the dimensions. The hour feature is as important as the temperature for the third dimension. Thus, the first dimension of the reduction captures more information about the usual hourly traffic and the other dimensions are more dependent on the weather properties.

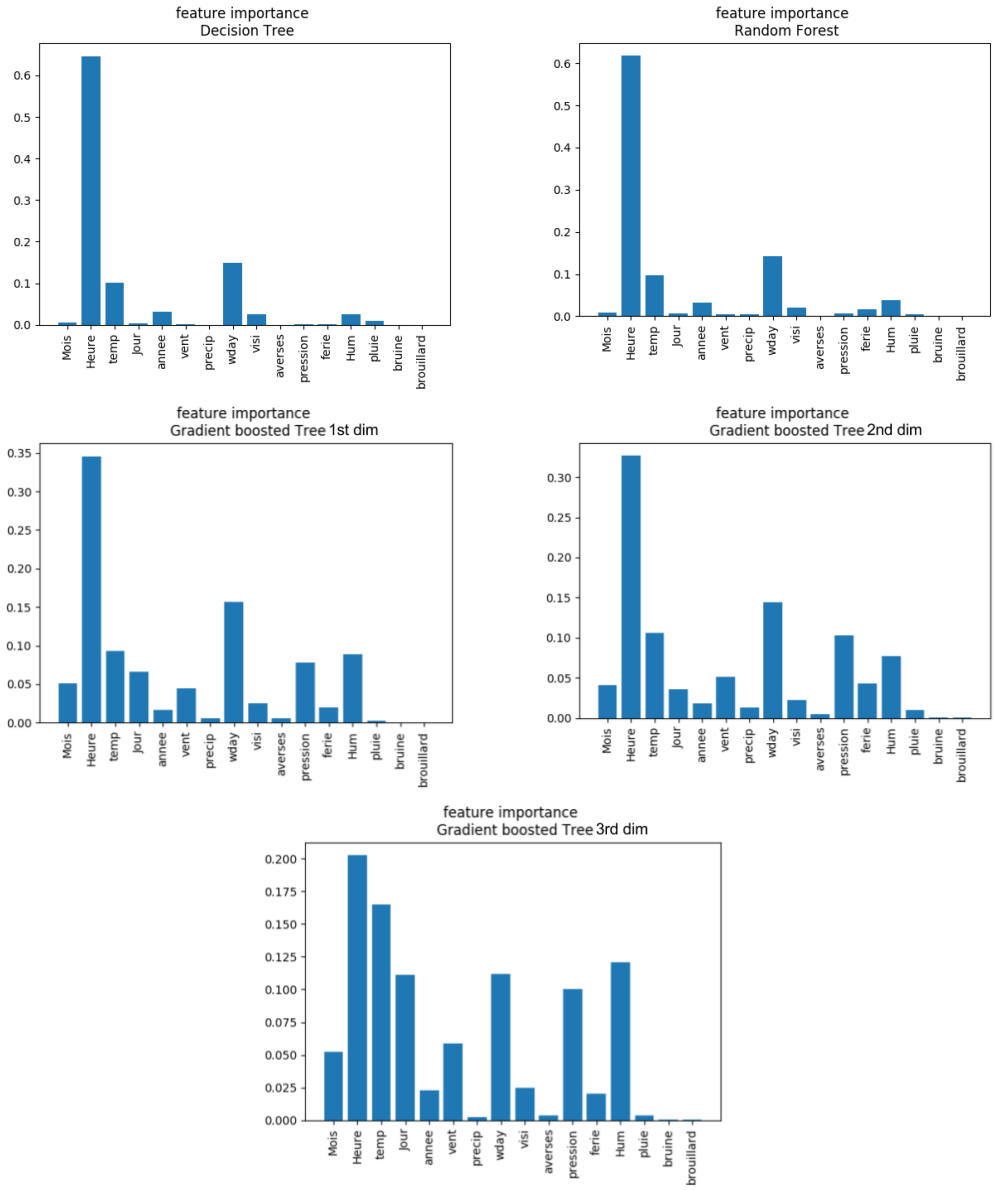

Figure 4.11 Feature importance for decision tree algorithms

4.5.3 Analysis of the results of the proposed algorithms

In this section we present some results, however, as the number of models is large, we present only the best results, and some selected ones. Tables are truncated.

Table 4.7 present the results for the best algorithms (RMSE). Results are ordered by increasing $RMSE$. Some algorithms do not appear in the table because their results are worse than the last one. Their results are displayed in appendix A. The algorithms are compared in terms of precision scores and training and prediction time. Training time is the computation time needed to learn the model, the testing time, the time to predict the validation set. These results are computed on the validation set. In the table GM refers to Gaussian Mixture, gbt to Gradient Boosted Tree, DT to Decision Tree and RF to Random Forest.

Table 4.7 Precision and running time of mean estimators

Reduction	Prediction	$RMSLE$	$RMSE$	MAE	R^2	$MAPE$	train(s)	test(s)
svd	gbt	**0.487**	**1.881**	1.022	**0.662**	0.427	15.109	0.147
autoencoder	gbt	0.487	1.902	**1.004**	0.654	**0.402**	10.783	1.783
svd	RF	0.492	1.925	1.039	0.646	0.443	2.407	0.293
autoencoder	RF	0.489	1.927	1.016	0.645	0.412	7.114	3.409
autoencoder	DT	0.498	1.981	1.042	0.625	0.423	6.321	3.988
svd	DT	0.499	1.99	1.06	0.622	0.446	1.199	0.088
GM	gbt	0.505	2.007	1.087	0.615	0.465	13.658	0.202
kmeans	gbt	0.505	2.006	1.088	0.615	0.464	13.159	0.165
id	gbt	0.499	1.951	1.047	0.641	0.434	753.54	17.58

There scores show that similar scores can be achieved with several reduction methods. The best scoring reduction method is the SVD. Good scores are also achievable with kmeans, Gaussian model or identity reductions and random forest or gradient boosted tree predictors. The SVD scores better than the identity reduction. It is also able to reduce significantly the training time. The SVD-gbt model is 50 times faster then the id-gbt one.
The next table (table 4.8) presents how well each reduction method performs with a linear prediction. The table is sorted per increasing $RMSE$. A gap is visible between naive methods (sum, dep-arr) and more complex ones and between efficient clustering, and less efficient (kmeans vs hierarchical). The conclusions made on the information conserved by the reduction algorithm are validated by these results : the loss of too much information, deteriorates the predictive capacity of the model. However even if the identity conserves all the information, it is not the best model. The SVD and kmeans reduction method are able to conserve the valuable information of the data and to recover main behaviors of stations.

These two tables compare the average performance of all algorithms on all stations, however,

Table 4.8 Comparison of reduction methods, using a linear predictor

Reduction	$RMSLE$	$RMSE$	MAE	R^2	$MAPE$	train (sec)	test (sec)
svd	0.602	**2.438**	1.326	**0.432**	0.612	2.099	0.083
GM	0.615	2.474	1.363	0.415	0.641	1.947	0.105
kmeans	0.615	2.475	1.364	0.415	0.642	13.453	0.101
autoencoder	**0.567**	2.518	**1.225**	0.394	**0.444**	3.515	1.344
sum	0.799	3.103	1.85	0.08	0.972	1.314	0.094
id	0.598	2.488	1.342	0.401	0.620	1.685	0.08

they do not capture all aspects of the problem. For example, some algorithms can be better than others for some stations. For the next results, the two best algorithms are computed for each station and a vote is given for each one of them. The Figure 4.12 gives the result of this vote. This figure shows that the best prediction algorithm is the gradient boosted tree, that scores well with all algorithms except the identity and the sum. The random forest also performs well. This figure also suggests that there is no reduction method that completely outperforms the others. This analysis shows that the clustering methods are adapted for some stations of the clusters, but outliers remain and should be learned in a different way. The SVD is able to better model these outliers. Therefore some algorithms are better suited for some stations. The Figure 4.13 presents the average station size per best algorithm. For each algorithm we compute the mean of the size of the station for which the algorithm is the best. This figure shows that some algorithm perform well on medium and big stations and some perform well on small stations. Therefore we can use the different algorithms for different stations. We build an ensemble model that can adapt the algorithm to the station.

4.5.4 Ensemble Model for mean estimation

The analysis of results showed that several algorithms score well. It also shows that they learn different information and score differently on stations. The first analysis showed that the best prediction algorithms were the gradient boosted tree and the random forest. It also showed that clustering is more adapted for some stations, and reduction methods, i.e., SVD or autoencoders are better on other stations. Hence we can use for each station the algorithm that scores the best. However, the high number of algorithms may introduce some overfitting, therefore we used only the best algorithm to build an ensemble method. We've selected the algorithms based on a SVD, kmeans, Gaussian Mixture or autoencoder reduction and on a random forest or a gradient boosted tree predictor. Then we use eight algorithms for the ensemble method.

We propose an ensemble approach, combining the bagging (Breiman, 1996) approach and

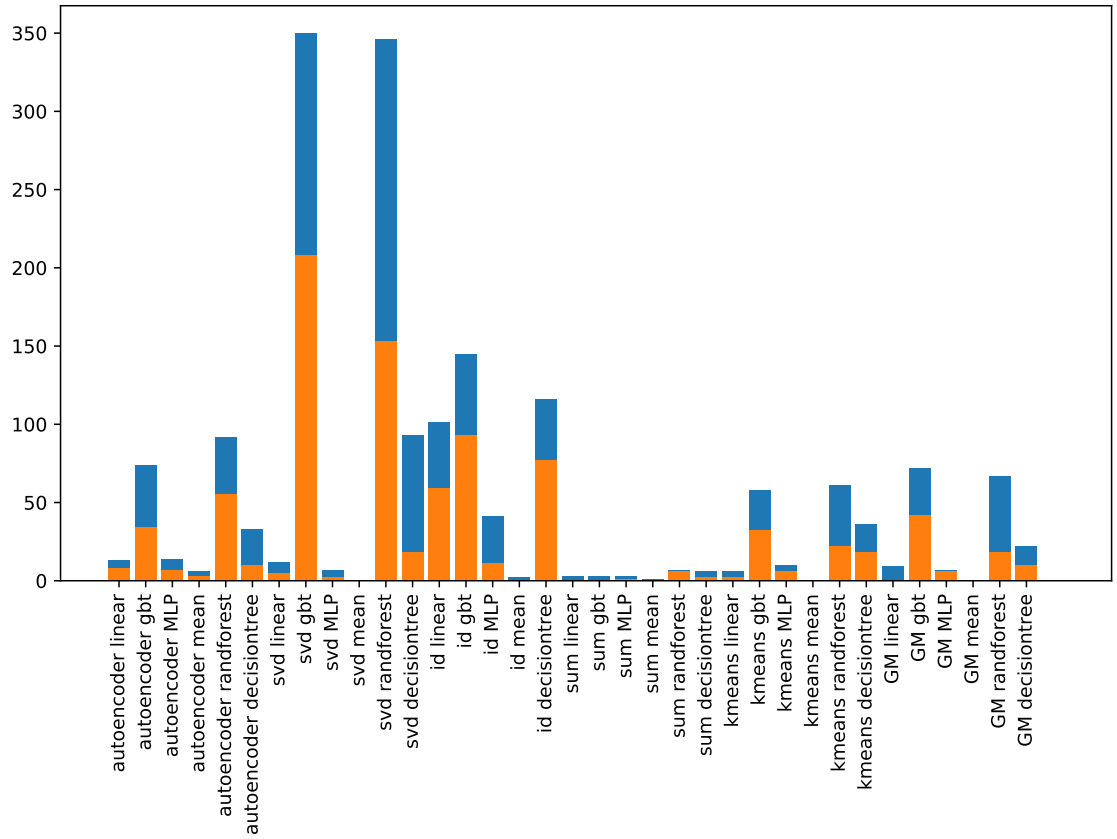

Figure 4.12 Number of stations for which each algorithms is the best

the bucket of models (Bensusan and Giraud-Carrier, 2000) approach. The proposed ensemble method selects for each station the n best models (reduction, prediction, inverse reduction) and averages their results. The bagging part is to average the results of n algorithms, the bucket of models part is to select for each station the set of n algorithms. As mentioned in the previous paragraph, to prevent overfitting, we limit the choice of the best model to a small subset of the possible ones. We name the obtained algorithm c_n, where n is the number of models used for the bagging operation.

To select the best model we trained all models on the training set, then we selected the best models for each station, based on the performance on the training set. Finally, we compare the performance of the algorithms on the validation set. Selecting the best algorithms on the result of the training set may introduce some overfitting. However results show that the ensemble method is effective. Therefore the selection of models does not overfit the training data.

Table 4.9 presents the precision results of the best algorithm, including the ensemble ones.

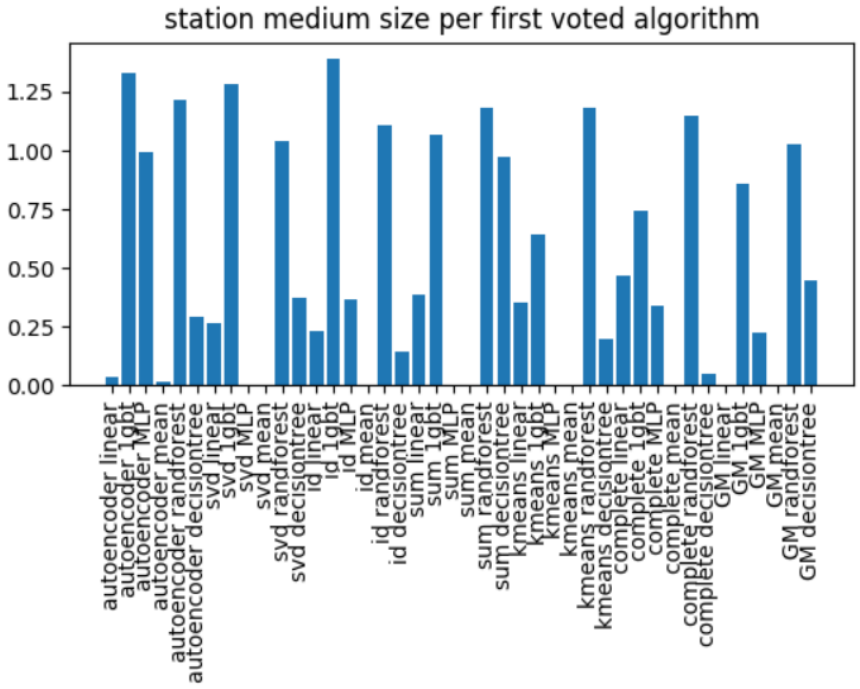

Figure 4.13 Medium size of stations per best algorithm

The table presents results sorted per increasing $RMSE$. Ensemble algorithms are able to outperform the previous ones. The best result was obtained using two or three algorithms per station. The ensemble model decreased the $RMSE$ from 1.882 to 1.868 and increased the R^2 score from 0.662 to 0.667. c_2 and c_3 models have very similar results, however, we noted that results on the c_3 model were more stable, then we used it for the this work.

The counterpart of this method is the increasing train and prediction times. The training time is the sum of train times of the eight algorithms for which stations can vote ($\sim$ 80 sec), and the prediction time is the sum of prediction times of these algorithms ($\sim$ 25 sec). The prediction time is displayed in the table. All five ensemble models have similar training and testing times.

Figure 4.14 is similar to Figure 4.12, but integrates the combined model (c_3) of the vote. Results of the new model are significantly better than the other algorithms.

Table 4.9 Results of combined algorithms

reduction	prediction	$RMSLE$	$RMSE$	MAE	R^2	$MAPE$	train (s)	test (s)
c_2	c_2	**0.484**	**1.868**	**1.015**	**0.667**	0.427	-	25.456
c_3	c_3	0.485	**1.868**	**1.015**	**0.667**	0.43	-	27.48
c_4	c_4	0.477	1.879	1.006	0.663	**0.412**	-	27.126
c_5	c_5	0.478	1.879	1.008	0.663	0.414	-	29.043
svd	gbt	0.487	1.882	1.022	0.662	0.427	13.774	0.219
c_1	c_1	0.488	1.883	1.015	0.661	0.419	-	26.452
svd	RF	0.49	1.914	1.035	0.65	0.441	2.301	0.518
id	RF	0.491	1.919	1.033	0.648	0.442	46.761	0.766
id	gbt	0.508	1.93	1.063	0.644	0.458	1112.7	5.369
AE	RF	0.498	1.984	1.032	0.624	0.406	13.68	10.23

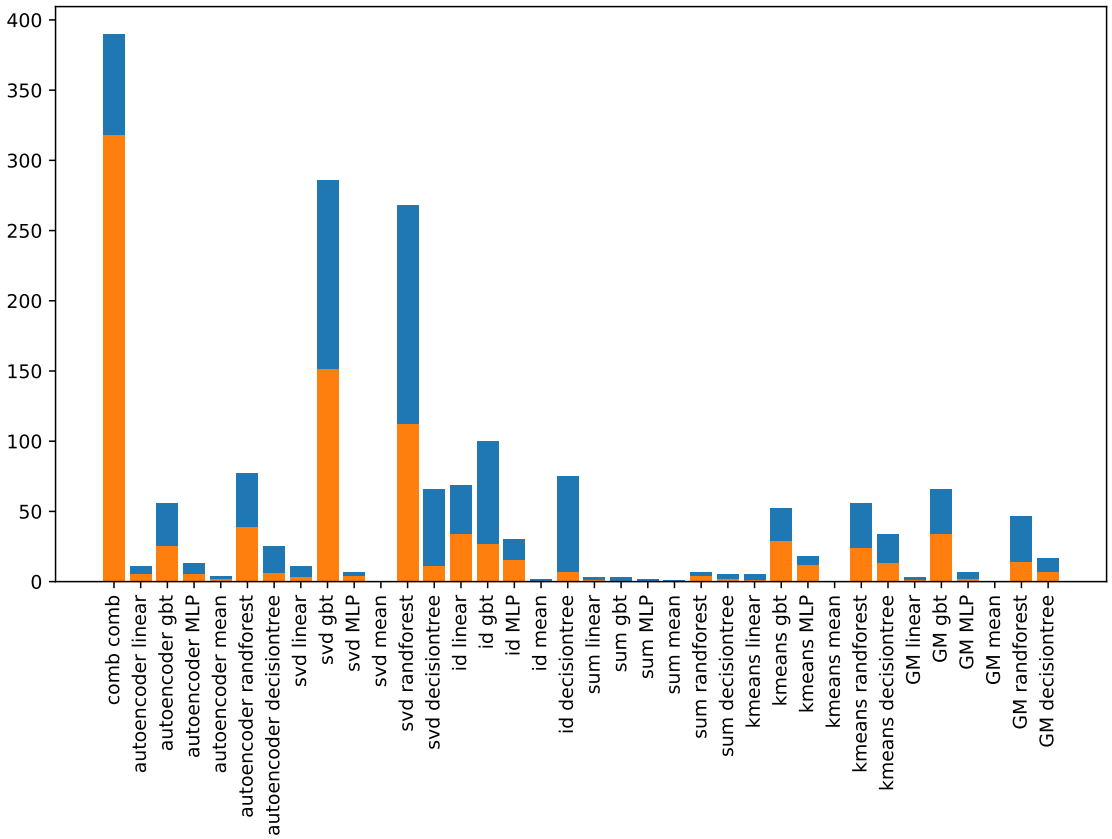

Figure 4.14 Votes for algorithms (orange first vote, blue second vote)

4.6 Predicting Traffic Variance

Once the traffic expectation is predicted ($\hat{\mu}_s(t, w)$), the next step is to predict the expected variance ($\hat{\sigma}_s(t, w)$) of the trip count per station. This estimation will be used to fit a distribution on the station that will complete the model. An overview on the residuals shows that the variance does not seem to be constant, especially regarding the hour, the temperature, the visibility, the humidity and the pressure. Then the variability of the traffic depends on the features. The Figure 4.15 displays the residuals versus the features. These graphs show some dependency between the error and the hour, the temperature, the wind, the week day, the visibility, the pressure, the humidity, the rain and the fog. Thus, learning an algorithm to quantify the confidence of the predicted value is justified. This observation was expected as the Poisson hypothesis, the most used one in the literature, supposes a variance equal to its mean. This section concentrates into learning and selecting a good predictor for the variance. This predictor will be used to fit a distribution on each station each hour, to complete the traffic model.

To estimate the variance we can note that the variance is the expectation of the squared error. Therefore a predictor trained on the squared error of the mean estimator, as we learned it on the trip count to estimate the mean, will estimate the variance. Instead of predicting the demand the new estimator predicts the squared error. However, the error of the estimator is much more random than the trip count and overfitting is more probable then, to avoid it we used only simple predictors. We also tried some complex prediction algorithms, but they all overfitted the data leading to very poor validation results. This estimator also learns data that is dependent on another estimator, then it is dependent on the performance of the first estimator.

This problem is similar to the mean estimation problem : we need to predict a matrix of the same size, where the trip counts are replaced with the squared error. However, as we try to predict an error, the data is much more stochastic, and most dependencies are not valid anymore. The stochasticity of this data does overfitting very easy, therefore simple models are used to limit it. An ANOVA (ANalysis Of VAriance) test on the features and residuals has been made and only significant features are kept : the hour of the day, the temperature, the pressure and the humidity. The week day is also used, due to its relationship with the hour feature. Several predictive models are learned to predict the variance. This analysis also proves that the constant variance hypothesis (H_0) is not correct. To estimate the variance, we trained :

— a linear predictor (and regularized ones)

— a decision tree

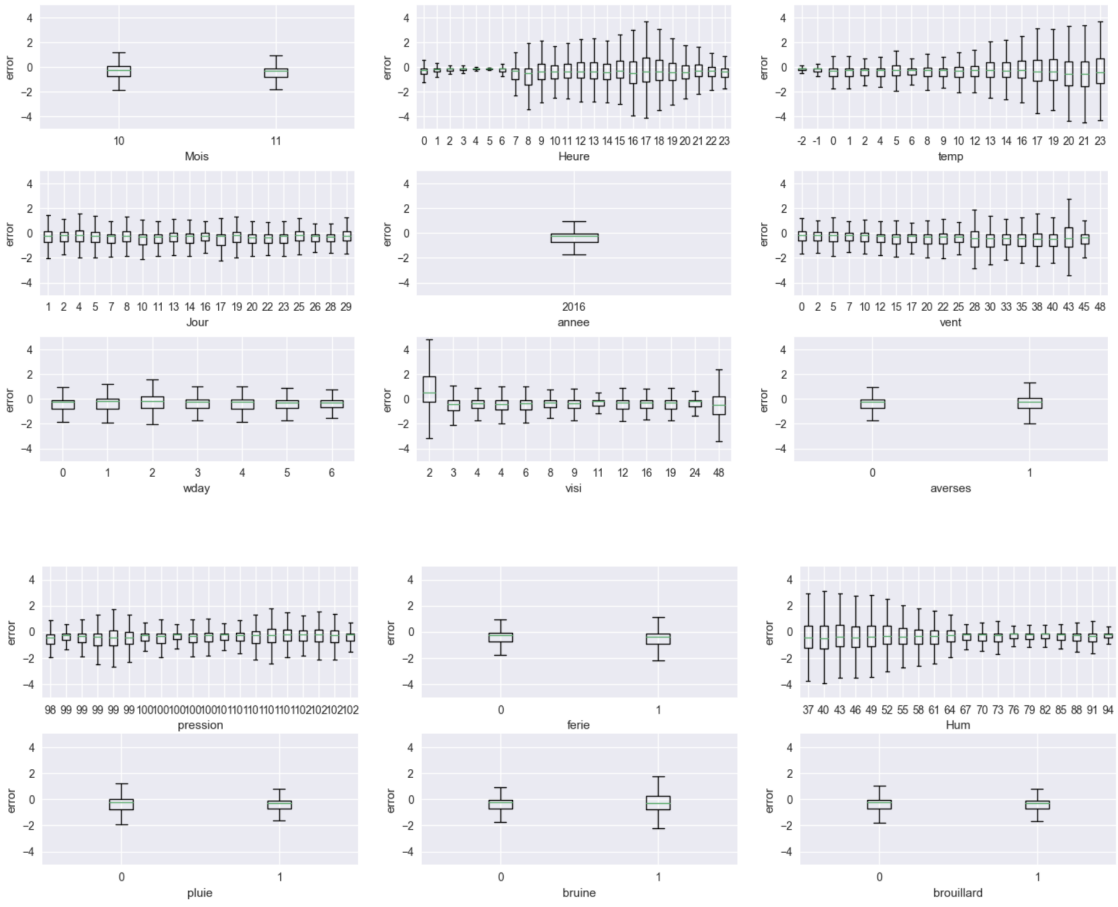

Figure 4.15 Residuals versus features

— a random forest

— a gradient boosted tree

Other predictors were judged too complex to model this stochastic problem. These predictors are trained to solve the following equation :

$$var(F) = (D - \hat{D})^2 \tag{4.20}$$

Where *var* is the variance estimator. However, this equation cannot be solved exactly (stochasticity) then we minimize :

$$(var(F) - (D - \hat{D})^2)^2 \tag{4.21}$$

The variance estimation problem is not well addressed in the literature, concerning bike-sharing systems, some articles may use it but none of them refers to it explicitly. The litera-

ture does not propose a methodology (for bike-sharing systems) to estimate the variance of stations. Most articles (Rudloff and Lackner, 2014; Chen et al., 2016; Gast et al., 2015) use the Poisson hypothesis that supposes a variance equal to the mean. For other distributions, articles rarely specify how the distribution parameters are obtained (Nair et al., 2013; Rudloff and Lackner, 2014; Gast et al., 2015; Gebhart and Noland, 2014).

Remark All distributions considered in the next section are overdispersed distributions ($\sigma > \mu$), then the variance estimation is corrected using the mean estimation : the variance is set to the maximum between the estimated variance and the estimated mean.

4.6.1 Variance hyperparameters optimization

Variance predictor hyperparameters have been optimized using the same approach (grid search) as for expectation predictors. We saw in the previous section that mean estimators are very close to each other and that they are able to encode almost the same information. Then we approximate the variance hyperparameters optimization problem, by considering these parameters independent of the mean estimator algorithm. We chose to optimize the variance estimator on the SVD-GBT model, as it is able to best explain the expectation of trips. These parameters have been validated for the other models, confirming this approximation.

The variance estimator is learned on the squared error of the mean estimator, therefore this data is very noisy. To avoid overfitting only simple models are learned. Some more complex models were tried and overfit the data. As outlined in Section 4.6 a feature selection has been done to limit the overfitting. To compare results, the variance is also computed using the common estimator without bias :

$$Var(D, \hat{\mu}_s(t,w)) = \frac{1}{|T||S| - 1} \sum_{t \in T, s \in S} (D_{t,s} - \hat{\mu}_s(t,w))^2$$

This estimator gives a variance value of 3.17 on the training data, the $RMSE$ score of this variance is then 3.14. We try to minimize in this section the $RMSE$ score, to approximate as precisely as possible the variance.

Linear Predictor

A linear predictor has been learned on the data. Categorical variables are used for the regression as they are better suited for linear models. We tested the usual regression, the Ridge regression and the Lasso regression. A grid search for the regularization parameter was done. The best parameter found on the Ridge regression is 360, and for the Lasso is 10.

Decision Tree based algorithms

Decision Tree A decision tree is also learned to predict the variance. It was optimized in terms of max depth (between 1 and 20) and minimum number of samples per leaf (between 20 and 1000). A grid search gave an optimal number of samples per leaf of 140 and the max depth of 10.

Random Forest The random forest algorithm was optimized in terms of max depth (between 2 and 20), minimum number of samples per leaf (20 to 1000) and number of estimators (5 to 100). The best configuration found had 40 estimators, at least 35 samples per leaf and a max depth of 11.

Gradient Boosted Tree The gradient boosted tree approach has been optimized on variance prediction in terms of the number of estimators (5 to 100), learning rate (10e-5 to 1) and max depth (1 to 20). The optimal parameters found use 8 estimators, a learning rate of 0.1 and a max depth of 3.

Table 4.10 presents the results of all algorithms, the best results were found with the random forest, with a $RMSE$ score of 2.813. This method uses bootstrapping to learn its trees. The bootstrapping method is very popular to estimate the variance as it helps to counter stochasticity. Some neural networks were also trained but had issues with overfitting.

Table 4.10 Performance of estimators on variance after optimization

algorithm	hyperparameter	$RMSE$
unbiased variance		3.17
linear regression		2.925
Ridge regression	alpha :360	2.847
Lasso regression	alpha :10	2.914
Decision Tree	min samp :140 max_depth :10	2.865
Random Forest	n_estim :40 max_depth :7 min samp :29	2.813
Gradient Boosted Tree	n_estim :8 max_depth :3 learning rate :0.1	2.873

4.6.2 Variance estimation results

After the optimization of variance predictors, we need to select the best one. We computed for the best mean estimation algorithms, the precision of each variance estimator. These results are displayed in table 4.11. These results show that the decision tree scores better than the random forest and gradient boosted tree. So, we can conclude that complex algorithms (random forest and gradient boosted tree) tend to overfit the data. However, experimentally we observed that these results are not stable when we change mean estimator (reduction and prediction). They found that the only algorithm with stable results is the linear regression. Then this algorithm is selected to estimate the variance, even if it is not the best one for each mean estimation model. This table has been computed on the validation data, then results slightly differ from the previous table.

Table 4.11 Precision of variance estimators (validation set)

Reduction	Prediction	Variance	$RMSLE$	$RMSE$	MAE	$MAPE$	R^2
svd	gbt	decisiontree	0.663	**8.754**	**2.980**	**0.861**	0.025
svd	gbt	gbt	0.682	8.852	3.022	0.911	0.020
svd	gbt	randforest	**0.657**	9.230	3.098	0.831	**0.033**
svd	gbt	linear	0.685	9.296	3.219	0.922	0.029
svd	gbt	var	0.761	9.869	3.436	0.981	0.005
svd	randforest	decisiontree	0.663	9.178	3.092	0.852	0.029
svd	randforest	gbt	0.684	9.316	3.148	0.910	0.027
svd	randforest	randforest	0.657	9.393	3.132	0.833	0.037
svd	randforest	linear	0.688	9.482	3.272	0.93	0.030
svd	randforest	var	0.762	10.03	3.479	0.987	0.007
autoencoder	gbt	decisiontree	0.671	9.935	3.274	0.833	0.023
autoencoder	gbt	gbt	0.682	9.942	3.315	0.856	0.022
autoencoder	gbt	randforest	0.668	9.842	3.260	0.843	0.02
autoencoder	gbt	linear	0.705	9.936	3.429	0.978	0.023

We can also see that best mean estimation algorithms give better variance estimator, the svd gbt model had better results for estimating the mean (Section 4.5.3), and the variance predictor is more accurate on it than on other mean estimation models. Estimating the variance is a delicate task and may lead to large errors. This variance prediction will be used to fit distributions on stations. Even if using specific estimators for each model may lead to better results, using the variance, gives a way to estimate parameters for all distributions.

4.7 Fitting the Trip Distribution

The last step to build the traffic model is to fit a distribution on the trip counts per station and hour. For this work we suppose some trip distributions, i.e. we test several functions $d(\mu, \sigma)$ of the model function $P(X_s = k|T = t, W = w) = d(\hat{\mu}_s(t, w), \hat{\sigma}_s(t, w))$. This section completes the model. Once this function is estimated the model will be able to estimate the probability $P(X_s = k|T = t, W = w)$ that k bike leave or arrive at time t at station s with a w weather. The function d is a distribution function defined in $\mathbb{N}$. Therefore we are going to explore usual distributions in $\mathbb{N}$:

— Poisson distribution

— Negative Binomial distribution

— Zero Inflated Poisson

These distributions are used in Rudloff and Lackner (2014); Gebhart and Noland (2014); Nair et al. (2013) papers. They showed that Negative binomial and Zero inflated distributions are better suited for this problem than the Poisson distribution. However they did not detail how the coefficients were estimated. In this section we compare the performance of each distribution and select the best one.

4.7.1 Comparing distributions : the Log Likelihood

In Section 4.2 we proposed some scores to evaluate the models, but we also said that they were not perfectly fitted for this task as they assume a distribution of the data. Gast et al. (2015) proved that precision scores are not well fitted to compare stochastic processes. These scores, also, do not consider the estimated probability distribution of trips. Therefore to compare distributions, we used the likelihood. This score compares the proposed distribution of trips to the reality. The likelihood is defined as the probability of reality, given the traffic model. Then getting the best model is equivalent to maximizing the likelihood. Let us note θ the traffic model, D the trip data random variable and d its real value (reality). M is the feature random variable and m its real value then :

$$\mathcal{L}(\theta, m, d) = P_\theta(D = d|M = m)$$

For repeated measures of an independent and identically distributed (iid) variable,

$$\mathcal{L}(\theta, m, d) = \prod_{t \in T} P_\theta(D = d_{t,\_}|M = m_{t,\_})$$

The Log-likelihood is usually used because it is more convenient to work with sums. Thus,

$$log(\mathcal{L}(\theta, m, d)) = \sum_{t \in T} log(P_\theta(D = d_{t,\_}|M = m_{t,\_}))$$

if we suppose that the trip count probabilities between stations are independent, the log likelihood is :

$$log(\mathcal{L}(\theta, m, d)) = \sum_{t \in T} \sum_{s \in S} log(P_\theta(D = d_{t,s}|M = m_{t,s}))$$

To get comparable results between tests, the normalized log likelihood can be used :

$$log(\mathcal{L}(\theta, m, d)) = \frac{1}{|T||S|} \sum_{t \in T} \sum_{s \in S} log(P_\theta(M = m_{t,s}|D = d_{t,s}))$$

The Log-likelihood is often used to support a hypothesis against another. It is a negative number. A log likelihood of zero means that the model represents perfectly the reality, that the probability of the reality given the model is one.

4.7.2 Poisson Distribution

The Poisson distribution is the most used distribution in bike sharing problem (Alvarez-Valdes et al., 2016; Raviv and Kolka, 2013; Nair et al., 2013; Rudloff and Lackner, 2014; Chen et al., 2016). The use of Poisson distributions is justified by the problem properties : Poisson distributions are suited for representing the number of times a random event occurs in an interval of time or space. In this problem, we model how many bikes are rented and returned in a specific station each hour. Arrivals and departures from stations are random and represent count data, therefore Poisson distribution seem suited to the problem. It assumes that :

— the number of occurrences is in $\mathbb{N}$

— the occurrence of one event does not affect the occurrence of others (events are independent)

— the rate of occurrences of events is constant

— two events cannot be simultaneous

These assumptions are mostly verified in the bike-sharing applications. However, some tricks and approximations are still necessary to use it. The rate of occurrence changes during the day in the bike-sharing problem. To solve this issue, the day is discretized in time steps of one hour. The rate of occurrences is considered constant during each time step. The other assumption that may be problematic is the independence of events : there are a lot of groups

of people that rent bikes, and then violates this assumption. The Poisson distribution is not able to model such behaviors and may lose in explanatory performance.

The Poisson model is characterized by its parameter λ. The mean and variance of the distribution are equal, and their value is λ. Therefore, the mean estimation is sufficient to fit this distribution to the data. The Poisson probability mass function is :

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

4.7.3 Negative Binomial Distribution

The negative binomial distribution is defined as the number of trials of a Bernoulli variable to achieve a defined number of successes. Two definitions exist, usually as the number of trials is at least the defined number of successes, only the number of failed trials is kept. The resulting variable is defined in $\mathbb{N}$. This distribution takes two parameters : the number of successes n and the probability of success p. The means of this model is $\mu = \frac{p.n}{(1-p)}$ and the variance is $\sigma^2 = \frac{p.n}{(1-p)^2}$. The probability mass function is :

$$P(X = k) = \binom{n+k-1}{k} p^k (1-p)^n$$

This distribution can be used as an alternative to Poisson distribution, especially with over dispersed data (data where the variance exceeds the mean). The negative binomial distribution is often used to model over dispersed Poisson variables. It is used in similar application than Poisson distributions, when the observed variance is greater than the theoretical one. In this work the two parameters are estimated using the estimated variance σ and mean μ, by the following formula $p = 1 - \frac{\mu}{\sigma}$ and $n = \mu\frac{1-p}{p}$.

4.7.4 Zero Inflated Poisson

The zero inflated Poisson is also an over dispersed Poisson Model, this model is characterized by an excess of zero in the distribution. The observed probability of zero is greater than it should be. Zero inflated Poisson distribution is defined by a combination of two distribution. With a probability ρ the model result is zero, and with a probability $(1-p)$ the model follows a Poisson distribution. This distribution can be more suited for instances where trips counts are not independent, justifying its use instead of a Poisson distribution. For example, if in a small station there is no traffic most of the time, and sometimes, a group of friends rent three or four bicycles, then these rentals are not independent, and the Poisson distribution is not suited in this special case. This phenomenon is not well represented by a Poisson distribution

that will assign a very small probability to this event, the zero inflated one in more adapted. The Zero Inflated probability mass function is :

$$P(X = k) = (1 - \rho)\frac{\lambda^k e^{-\lambda}}{k!} + \mathbb{1}_{k=0}\rho \tag{4.22}$$

Where $\mathbb{1}$ is the characteristic function. This distribution has two parameters : the Poisson parameter λ and probability ρ. The mean of the distribution is $\mu = (1-\rho)\lambda$ and the variance $\sigma^2 = (1 - \rho)\lambda(1 + \rho\lambda)$. To estimate the parameters of the distribution we compute : $\lambda = \mu + \frac{\sigma^2}{\mu} - 1$ and $\rho = \frac{\sigma^2-\mu}{\sigma^2+\mu^2-\mu}$. The property of variance and means of distribution on $\mathbb{N}$ make these estimates admissible.

4.7.5 Distribution Results

We saw in Section 4.5 which model is better to predict data with different measures. These tests tried to predict as precisely as possible traffic expectation. To complete the model, we must fit a distribution per station and hour and compare the likelihood of each model. We compared the results of the three distributions presented in the previous section with several mean estimation models on the validation dataset.

We compute the likelihood of each station at each hour then we aggregate it in several ways. First to compare distribution hypotheses (table 4.12), we average the likelihood over the stations and hours. Then we compare the distribution hypotheses per station (Table 4.13 and Figure 4.16), then we average the log likelihood over the hours.

The table 4.12 shows the average likelihood of some mean estimation models and a linear variance prediction model. We selected only models with a good performance to compute the log likelihood. The Poisson distribution hypothesis gives the best results for almost all algorithms, but this result is an average, a finer analysis can give more insight on the behavior of stations.

The Figure 4.16 is built similarly as Figure 4.12. Each station gets three votes, one red (first), one green (second), and one blue (third), and gives it to the three models (reduction + prediction + distribution) that have the best likelihood on the station. The x axis represents a selection of models (best ones) represented by the reduction method, the prediction method and the distribution hypothesis (P = Poisson, ZI = Zero inflated, NB = Negative Binomial). We recall that the $c3$ predictor is the ensemble model that uses the three best algorithms for each station. The y axis represents the number of votes for each algorithm. This figure shows that the combined model built in the Section 4.5.4 has very good results and that most

Table 4.12 Average Log likelihood per distribution hypothesis of several models

Reduction	Prediction	Negative Binomial	Poisson	Zero Inflated
autoencoder	gbt	-1.684	-1.184	-1.572
autoencoder	randforest	-1.583	-1.091	-1.75
svd	gbt	-1.245	-0.988	**-1.425**
svd	randforest	**-1.218**	**-0.959**	-1.444
kmeans	gbt	-1.249	-0.982	-1.55
kmeans	randforest	-1.25	-0.978	-1.542
GM	gbt	-1.258	-0.984	-1.571
GM	randforest	-1.244	-0.975	-1.573

votes went to the Poisson hypothesis (last third of x axis). The two other hypothesis also get some votes, which are displayed in table 4.13. This table shows for each distribution the average size of stations (section 4.2) that first voted for the distribution, and the percentage of first, second and third votes they got (best, second best and third best). It shows that the Poisson hypothesis fits best most stations. The two other distributions get only 5% of the votes each. This proportion of stations preferring a zero inflated or negative binomial distribution is not sufficiently high to justify an ensemble method. This result may be due only to stochasticity of the results and may not represent reality. However, a clear trend is visible in these stations : as expected stations that prefer zero inflated distributions are small (0.30 trips per hour on average) and stations that prefer Negative Binomial distributions are quite big (2.13 trips per hour on average). This conclusion may help build a more precise model, by selecting for small stations the zero inflated distribution and the negative binomial one for big stations. However, a further analysis has shown that the zero inflated distribution is as suitable for small stations that the Poisson one and that the Negative Binomial is as likely as the Poisson Distribution for big stations.

This distribution analysis concludes that the Poisson distribution is more adapted to model the demand. However, these results have to be mitigated. The variance prediction is learned on the error of a first predictor, and may be inaccurate. Therefore it decreases the score of zero inflated and negative binomial distributions.

Table 4.13 Proportion of votes per hypothesis

distribution	mean station size	% 1^{st} votes	% 2^{nd} votes	% 3^{rd} votes
Negative binomial	2.13	4.8	4.5	4.8
Zero inflated	0.30	5.6	6.7	7.5
Poisson	0.92	89.6	88.6	87.6

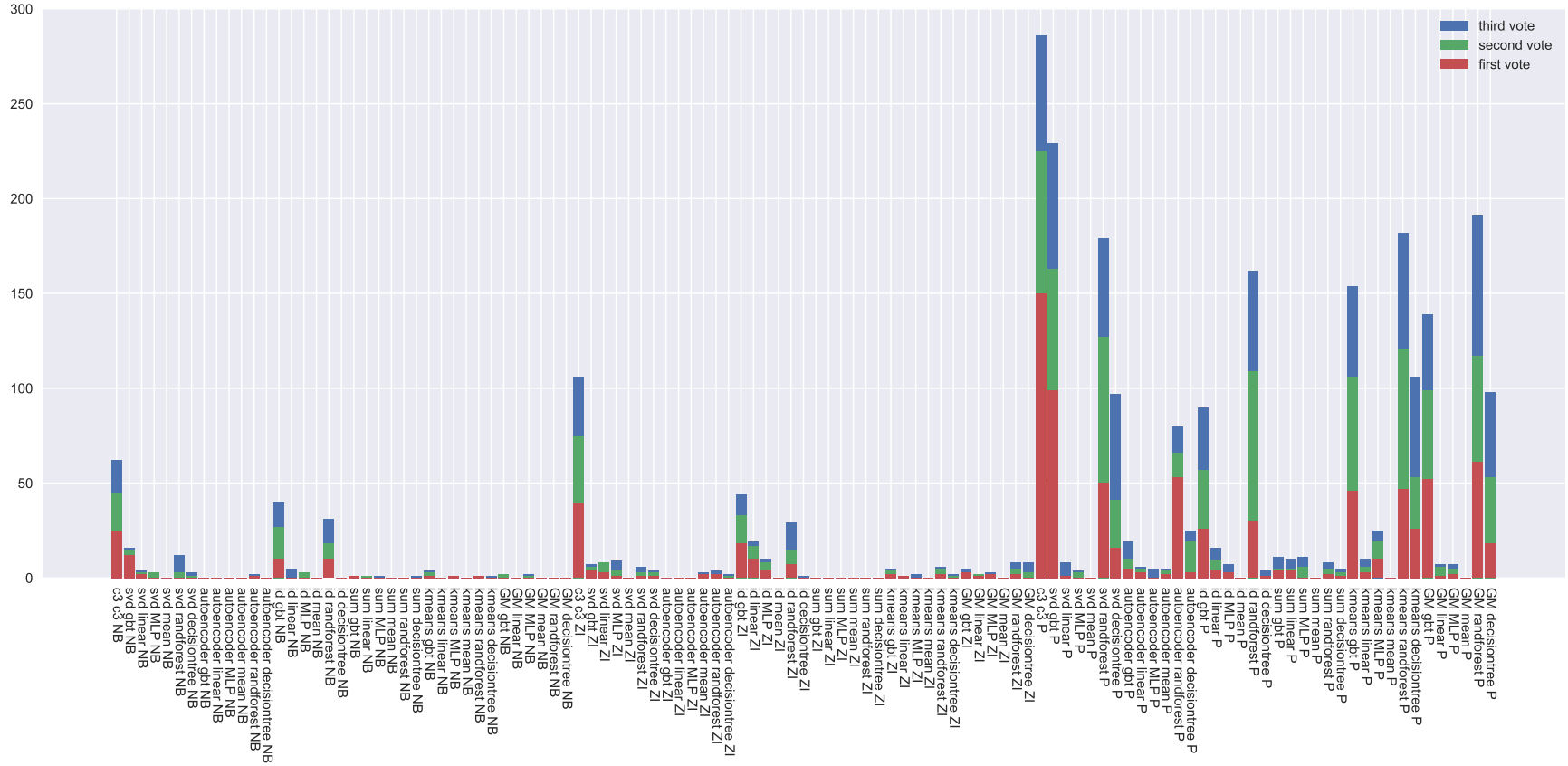

Figure 4.16 Votes for best algorithms, based on the best *RMSE*

4.8 Results on the Test dataset

To validate the results of previous sections, the models were trained on the whole training and validation set (data from the 01/01/2015 to the 30/09/2016). This section compares the scores of the main algorithms on the test set. As previous results were used for making some decisions, and to compute the ensemble model, some overfitting is possible. Then we test the generalization capacity of our models on a new test sets. This section also applies this same methodology to New York and Washington systems.

4.8.1 General Results

Table 4.14 presents the score of each algorithm, ordered by $RMSE$. The mean log-likelihood was also computed using a Poisson distribution hypothesis. This table shows that the best algorithm is the ensemble one. However, the model using a SVD reduction and a gradient boosted tree predictor has very similar scores. These results agree to the one of the previous sections : SVD, Kmeans and Gaussian Model reduction methods have very similar results, and the ensemble model is able to outperform them. However the improvement achieved by the ensemble model is not as good as in the validation set. The svd-gbt model is able to get better score than the ensemble model.

The SVD approach achieve the best results using only 10 predicting dimensions, reducing the size of the problem of 99%. Therefore the SVD is able to keep most of the relevant information. As the SVD performs better than the identity reduction, then it is able to erase some noise of the data. Clustering algorithms stabilize after 10 clusters and are not able to recover as much information as the SVD. The autoencoder has slightly better results than the SVD, in terms of conservation of information (RL), but reduced dimensions are harder to predict and to train. It remains competitive and achieves good results.

Figure 4.17 represents the deviation of errors after the last training example. It displays the average error on each time step for each score. Labels on the horizontal axis are spaced one week apart. This figure shows that even if there are some hard to predict days, the error stays constant or tends to decrease. The decreasing error is mostly explained by the decreasing number of trips during the testing period. Week ends are visible because of the improving scores except for the R^2 that deteriorates. This behavior can be explained by the decreasing number of trips, and their distribution during the day. The week end at time step 168 last one day more than others, because of the Thanksgiving day. The week end at time step 504 behave strangely as scores are particularly good, even the R^2 improves. The model is able to detect exceptional events. This event detection could be used to improve

the performance of the algorithm. The figure also compares the performance of the SVD and kmeans reductions. This figure also illustrates that the SVD model better the problem. The SVD curve is always over the kmeans one.

These graphs show also that the SVD GBT model and the SVD random forest models are almost equivalent. Both can learn more effects than kmeans based algorithms. All scores tend, at the end of the year to stabilize at a medium score ($MAPE$ at 0.33, Loglikelihood at 0.9, $RMSE$ at 1, $RMSLE$ at 0.37 and R^2 at 0.60). The R^2 discriminates effectively the models.

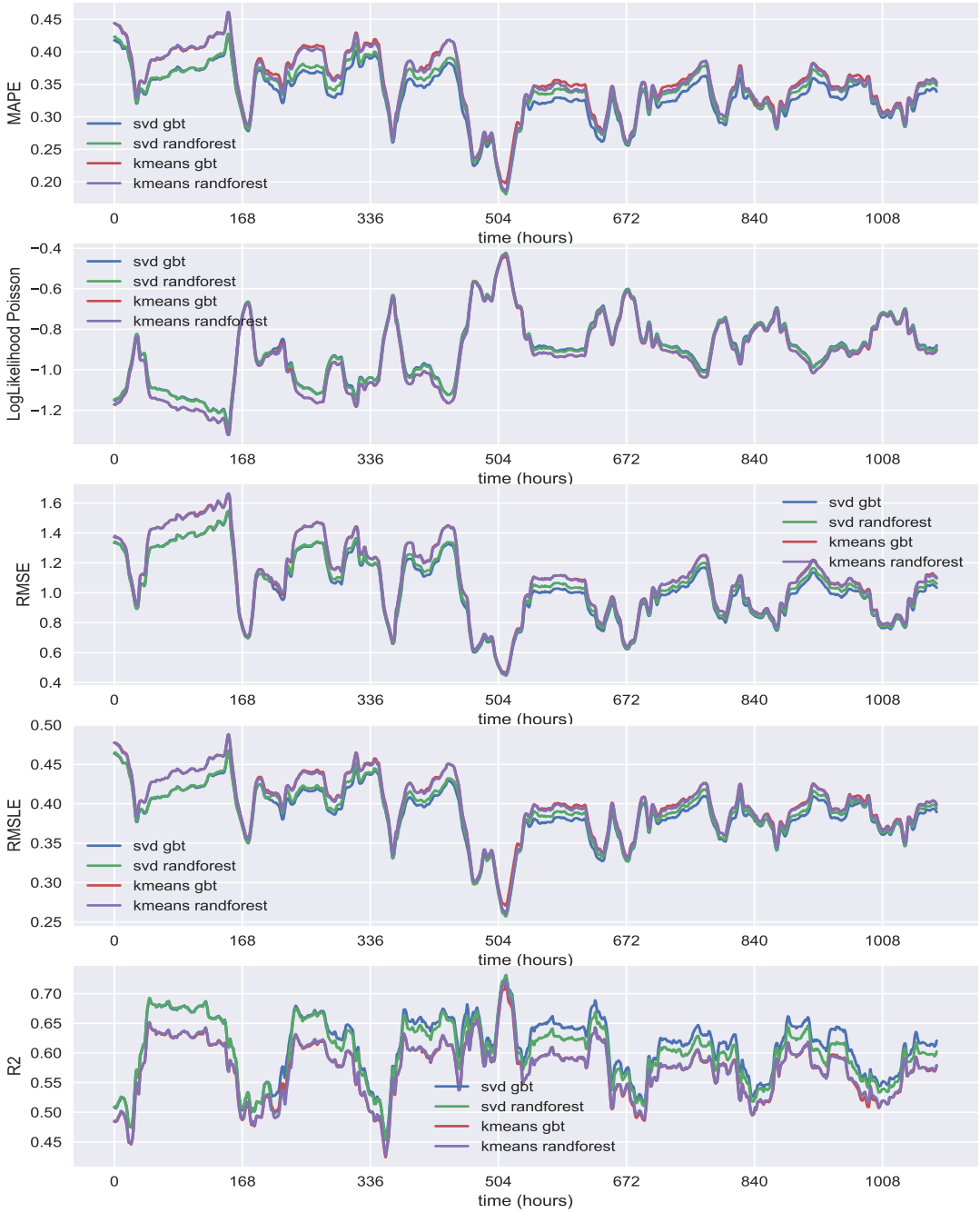

Figure 4.17 Error Deviation

Table 4.14 scores of each model on the test set

algorithm	$RMSLE$	$RMSE$	MAE	R^2	log likelihood	$MAPE$	train (sec)	test (sec)
svd-gbt	**0.424**	**1.268**	**0.65**	**0.624**	-1	**0.323**	20.84	0.11
$c_4 - c_4$	0.434	1.28	0.681	0.617	**-0.936**	0.369	0	13.794
$c_3 - c_3$	0.435	1.281	0.682	0.616	-0.939	0.371	0	11.979
$c_2 - c_2$	0.434	1.283	0.682	0.615	-0.944	0.369	0	11.797
$c_5 - c_5$	0.434	1.285	0.683	0.614	**-0.936**	0.371	0	11.373
svd-randforest	0.436	1.29	0.695	0.611	-0.942	0.379	2.944	0.265
kmeans-gbt	0.442	1.378	0.696	0.556	-1.031	0.35	19.641	0.106
autoencoder-gbt	0.455	1.42	0.698	0.529	-1.106	0.354	19.33	5.242
GM-gbt	0.443	1.373	0.699	0.56	-1.018	0.356	19.155	0.132
$c_1 - c_1$	0.451	1.352	0.712	0.573	-0.985	0.389	0	15.736
svd-decisiontree	0.444	1.355	0.713	0.571	-0.973	0.38	1.719	0.072
kmeans-randforest	0.443	1.375	0.721	0.558	-0.966	0.385	3.343	0.27
kmeans-decisiontree	0.451	1.408	0.733	0.536	-0.99	0.386	1.724	0.062
GM-randforest	0.462	1.405	0.754	0.539	-0.983	0.423	3.188	0.273
autoencoder-randforest	0.481	1.464	0.762	0.499	-1.05	0.427	11.729	5.498
autoencoder-decisiontree	0.498	1.554	0.803	0.435	-1.088	0.452	10.578	5.207
GM-decisiontree	0.499	1.506	0.821	0.47	-1.036	0.479	1.739	0.071
autoencoder-linear	0.541	1.853	0.822	0.197	-1.801	**0.319**	9.078	4.414
svd-linear	0.516	1.662	0.844	0.354	-1.248	0.441	2.174	0.068
GM-linear	0.519	1.682	0.853	0.339	-1.25	0.446	2.37	0.063
kmeans-linear	0.52	1.684	0.854	0.337	-1.249	0.448	14.431	0.072

4.8.2 Results per stations : stations 6507, 5005, 6221

The previous section presented some results on the models. These results were averaged and are therefore difficult to interpret in some cases. This section will focus on some specific stations. Figure 4.18 presents for each station its error ($MAPE$) compared to its size (average number of trips per hour). This error first increases, until an average size of 1. Then it stabilizes at an error of 0.4. There are two scores per station, one for the arrivals and one for the departures.

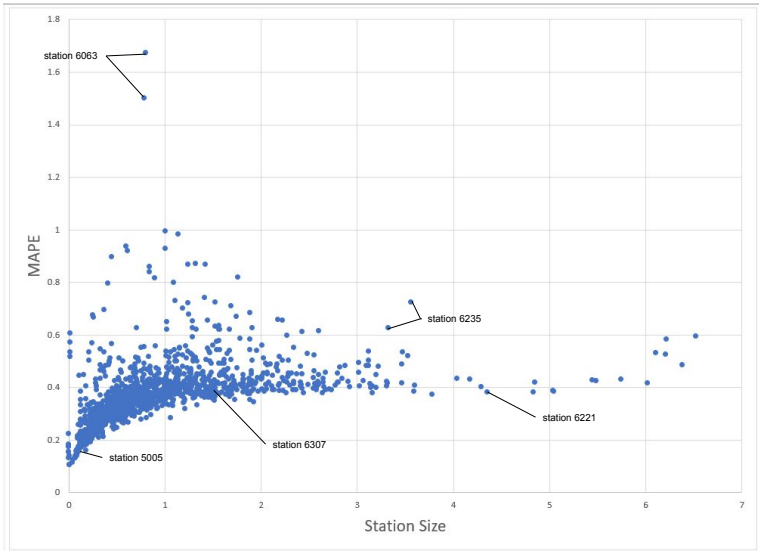

Figure 4.18 Station size versus MAPE

This graph also highlights some outliers. These outliers correspond to stations that suddenly changed behavior. For example, the station 6063 was closed 75% of the test time and station 6235 changed behavior between 2015 and 2016, doubling its traffic. Station 6043 behavior changed completely at test time as the employee in charge of removing bikes in that particular station was not effectively performing his duty. The model was not able to infer this change.

Figure 4.19 shows the $R2$ scores versus the size of the station : the bigger the station, the better its behavior is explained. Some outliers are also visible and corresponds to station that have been shut down during the test period.

We will now analyze the model on three different stations of various sizes. We selected Station 6221, Station 6307 and station 5005. They are shown in Figure 4.18 and in Figure 4.19. These

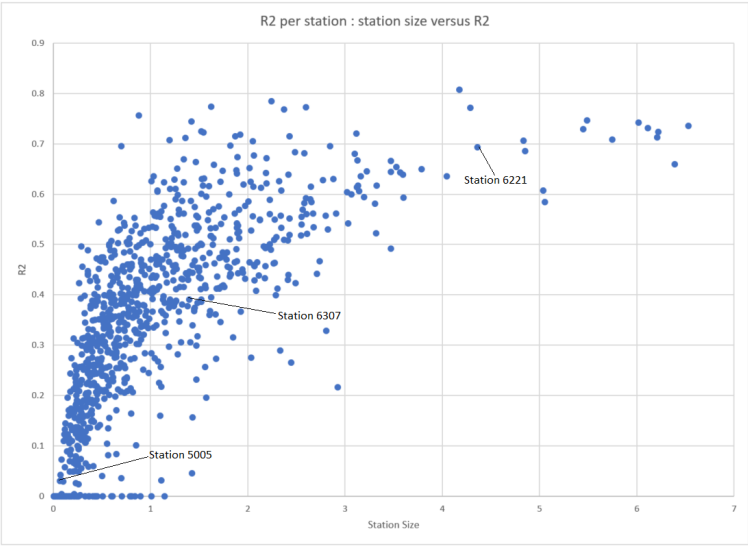

Figure 4.19 Station size versus R2

stations are representative of the diversity of stations. Station 6221 is a big station ($4.35 > 2$ trips per hour), station 6307 is a medium station ($2 > 1.52 > 0.5$ trips per hour) and station 5005 is a small station ($0.07 < 0.5$ trips per hour). Table 4.15 presents scores on these three stations. This table shows that the bigger the station, the bigger the error (as presented in Figure 4.18). However, the R^2 score mitigate this conclusion as only 4.1% of the small station variance was explained whereas 70% of the big stations variance is explained. Therefore the bigger the station, the more its behavior is explained. Below, two models will be compared with these stations. The first model uses a SVD reduction and a gradient boosted tree prediction. The second one uses a Gaussian model clustering and then a gradient boosted tree prediction.

Table 4.15 Scores on three stations

Station	Size	$RMSLE$	$RMSE$	MAE	R^2	Log likelihood	$MAPE$
Station 6221 rentals	4.36	0.463	2.455	1.672	0.692	-2.004	0.380
Station 6307 returns	1.52	0.493	1.495	0.994	0.391	-1.491	0.384
Station 5005 returns	0.07	0.195	0.318	0.169	0.041	-0.265	0.133

Station 6221 (rentals)

This station is a big station of the network, with about 4.35 trips per hour on average and up to 36 trips in one hour. This station makes quite big errors but has a R^2 score of approximatively 0.7. Figure 4.20 presents the difference between predictions and real traffic

for two prediction models (SVD GBT and GM GBT). A clear pattern is visible and learned by the model. However, the model tends to underestimate peak hours.

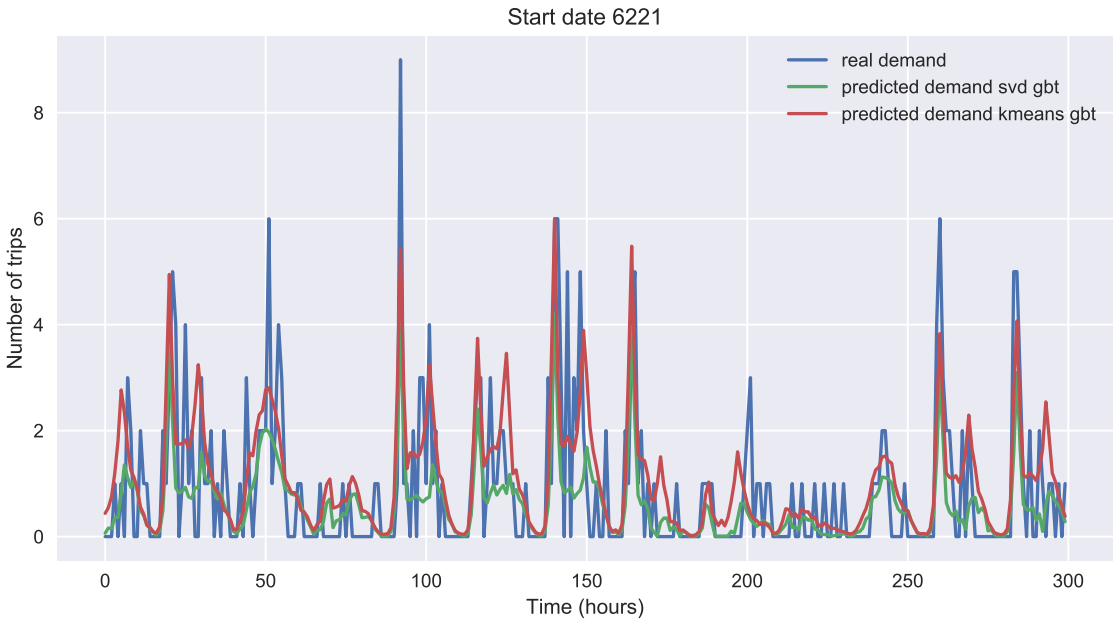

Figure 4.20 Predicted and real Traffic in station 6221

Station 6307 (returns)

This station is a medium station with about 1.52 trips per hour. It has a maximum of 13 trips per hour. It scores are close to the mean score of the network. Figure 4.21 shows the difference between prediction and real demand. The real demand is quite chaotic, and the predicted demand proposes a good estimation of the trip expectation. Peak hours are also underestimated.

Station 5005 (returns)

This station is a small station with about one trip per 14 hours. It has a maximum of four trips in one hour. It has very good scores, due to its small number of trips. A minimum trip expectation of 0.1 is imposed on all stations all hours, therefore the predicted demand tends to be overestimated. Figure 4.22 presents the predicted and real demand for the SVD GBT and kmeans GBT models. The real demand is very unpredictable and irregular. However, the model is able to learn the position of some recurrent trips. The trip expectation is distributed around actually observed trips. On these stations, only trips that happen regularly

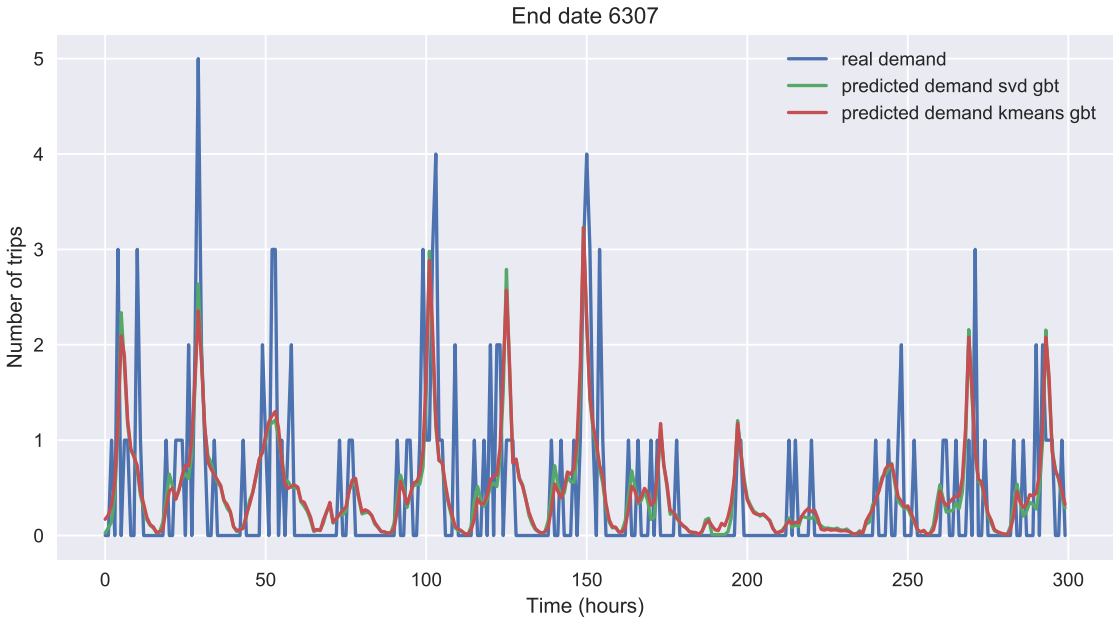

Figure 4.21 Predicted and real Traffic in station 6307

are predictable. Then for the small stations, only commuter's trips are modeled.

Figures 4.20, 4.21 and 4.22 show that the SVD outperforms the clustering reduction giving more predictions closer to reality than the Gaussian model. This analysis also showed that bigger stations are more predictable than smaller ones even if the resulting error is bigger. The MAPE error stabilizes at 0.4, for stations bigger than 1 trip per hour.

4.8.3 Geographical distribution

Figure 4.23 presents geographically the average error (MAE and R^2) on stations. This MAE error is concentrated in the "Plateau", and downtown, as they regroup most of Montreal bicycle activity. Peripheral stations have fewer trips and therefore less error (as seen in previous subsection). The R^2 score plots show that stations with a R^2 score higher than 0.55 are in downtown and in the "plateau". This result confirms the one of the previous subsection : Bigger stations are better more explained even if they have a bigger absolute error.

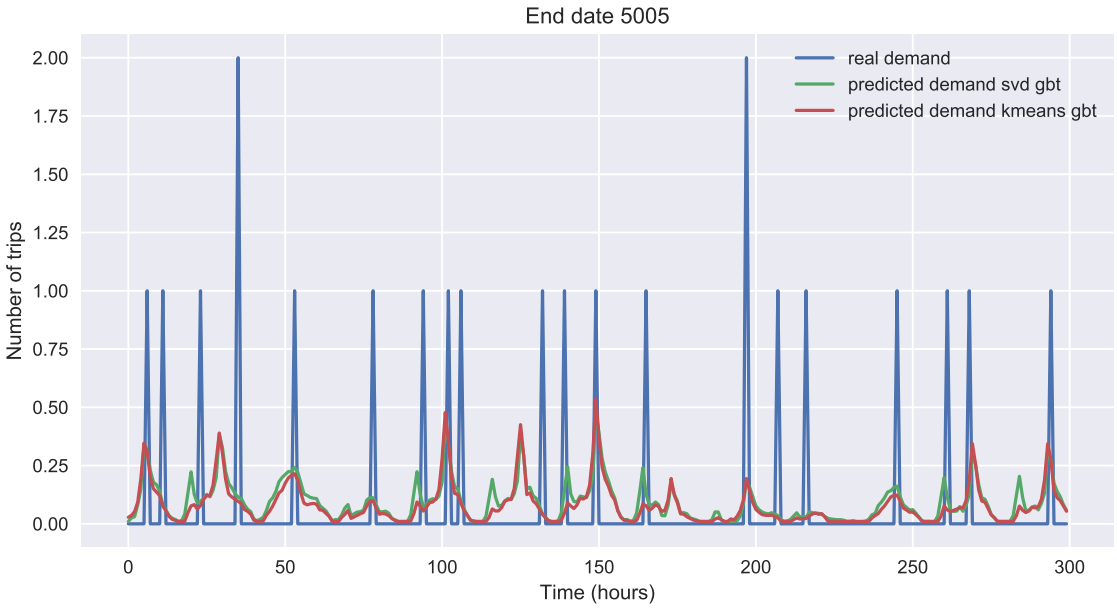

Figure 4.22 Predicted and real Traffic in station 5005

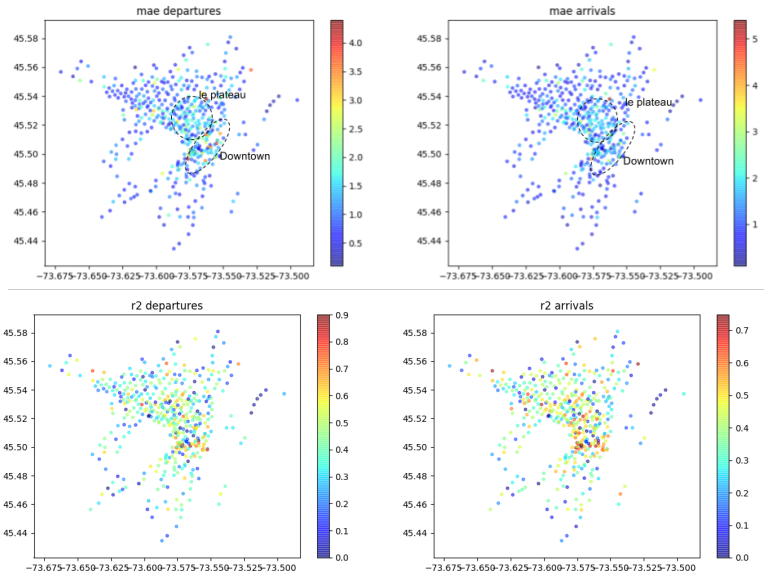

Figure 4.23 Geographical distribution of the error

4.8.4 Adding new stations

Every bike-sharing system change over time, by adding or removing stations. The reduction part of the model makes the addition of a new station quite easy and is able to predict its behavior with a small amount of information. The model architecture modelizes the expectation of these main behaviors, and the inverse reduction function rebuilds from these behaviors the station expectation. This inverse reduction function is for this work, deduced from the inversion of the reduction method. Adding a new station or changing the behavior of a station is therefore equivalent of finding for most methods (linear ones) the best set of parameters to express the station expectation as a linear function of the extracted behaviors. The best parameters can be estimated using a linear regression on this coefficient. This linear regression takes the extracted behavior predictions as features and the station trip count history as an objective. This method can be used to add stations, but also to refine station behaviors, during the year.

4.9 Application on other bike-sharing networks

Previous sections and chapters built a demand model for bike-sharing systems using Montreal's data. To validate the conclusions of this work, we trained our models on data from New York and Washington. We learn the best models selected with Montreal data and validate previous conclusions on other networks. These models are trained only on temporal features (without weather features), this section aims to compare the approaches, not to show the improvement gained using the weather data.

We retrieved trip counts data from the operator's website and preprocessed it as Montreal trip data. As weather data is not used, no missing data completion was needed. The following table shows the best scoring algorithms. The Appendix B shows the complete result tables. We recall that c3 is the ensemble method.

4.9.1 Montreal

The models were retrained on Montreal data without the weather features to be able to compare the performance of the models between networks. Networks are then compared with the same conditions, then they can be compared. Table 4.16 presents these results. We can conclude that previous remarks are still valid : the SVD reduction works the best, reducing the problem and maintaining most information efficiently. The GBT and RF are the best predictors. The removal of weather features didn't change the results. However, scores are a bit worse than previously, because of the absence of weather variables. The R^2 scores

decrease from 0.62 to 0.48, and the $RMSE$ goes from 1.268 to 1.48.

Table 4.16 Scores on Montreal (Bixi) without weather features

Algorithm	$RMSLE$	$RMSE$	MAE	R^2	Log Likelihood	$MAPE$
svd, gbt	0.494	1.483	0.804	0.486	-1.035	0.469
kmeans, gbt	0.5	1.524	0.817	0.457	-1.059	0.472
GM, gbt	0.504	1.527	0.825	0.455	-1.061	0.482
autoencoder, gbt	0.499	1.558	0.8	0.432	-1.091	0.449
GM, randforest	0.474	1.584	0.795	0.414	-1.025	0.427

4.9.2 Washington

The Trip data of Washington is available on their website. Data from 2015 and 2016 was retrieved and cleaned (missing station information, changing station names,...). Washington Network is composed of 499 stations and work all the year. 6.5 million trips were done in two years. The training matrix is composed of 998 columns and 15 thousand lines. This data is slightly bigger than Montreal one. Table 4.17 presents the results for Washington's network. The conclusions made for Montreal systems are still valid : The combined algorithm is the best, but its result is slightly better than the one of a SVD reduction and a gradient boosted tree predictor. SVD, Gaussian Model and kmeans reduction methods have similar performance. Gradient boosted tree and random forest also have good performance. The performance achieved on Washington's system without weather data is slightly similar to the one achieved by Li et al. (2015) ($RMSLE = 0.35$). Li et al. (2015) proposed a model using geographical clustering for traffic prediction, and used data from years before. However they used weather features to improve their results, then our approach gives similar results than theirs.

Table 4.17 Scores on Washington (Capital Bikeshare)

algorithm	$RMSLE$	$RMSE$	MAE	$MAPE$	R^2	LL
$c_3 - c_3$	0.380	1.268	0.530	0.252	0.605	-0.768
SVD-GBT	0.386	1.27	0.532	0.254	0.601	-0.828
SVD-random Forest	0.383	1.278	0.544	0.279	0.596	-0.751
SVD-Decision Tree	0.395	1.342	0.560	0.286	0.554	-0.793
id-GBT	0.396	1.294	0.575	0.294	0.582	

4.9.3 New York

The trip data of New York was available on their website. We retrieved the trip data from 2015 and 2016, and their station information (location, capacity, names,...). New York bike-sharing system uses 812 stations. They don't close the network during winter. The collected data represents 15 thousand hours and 22.5 million trips. This data is significantly bigger than Montreal data, this difference makes the learning quite difficult as it exceed the available memory of the computer. The reduction method simplifies the problem such that it became tractable on an usual laptop. To predict be able to predict on this network without reducing it is necessary to predict one station at a time. Table 4.18 presents the results of the New York bike-sharing system (CityBike). The conclusions are the same as for Montreal and Washington. The combined approach is able to improve the R^2 score from 0.42 to 0.5. This improvement is significant.

Table 4.18 Scores on CityBike (New York)

algorithm	$RMSLE$	$RMSE$	MAE	R^2	LL	$MAPE$
c_3,c_3	0.586	3.097	1.247	0.442	-1.623	0.436
svd, gbt	0.624	3.106	1.267	0.439	-2.43	0.398
svd, randforest	0.586	3.206	1.323	0.402	-1.584	0.532
svd, decisiontree	0.604	3.262	1.38	0.381	-1.585	0.612
svd, linear	0.848	3.944	2.039	0.095	-2.204	1.146
autoencoder, gbt	0.952	4.485	1.686	-0.17	-inf	0.361
autoencoder, randforest	0.955	4.501	1.695	-0.178	-inf	0.367

4.9.4 Conclusions

The analysis made in the previous sections is still valid on other networks. The SVD reduction is very efficient : very fast to compute, conserves a high proportion of the data and needs only 10 dimensions. The reduction methods are very efficient in reducing the problem scoring better than usual methods. A flexible reduction as the SVD is able to adapt to every station, outperforming the kmeans clustering in explaining the traffic. Models with no reduction are outperformed by the one using the reduced problems. These problems are able to extract all or most of the valuable information and leave the irrelevant one. Building a model per station is therefore inefficient and tends to overfit the data. Reduction improves the precision of the model, and the training and testing time. This reduction also allows the model to quickly learn the behavior of a station, with less than one month of data. It is able to predict its behavior for the rest of the year. An ensemble approach is able to improve on

the score of the predictors, however, this difference is not significant and training and testing times are greatly increased. This chapter built an efficient model using a problem reduction, outperforming state of the art predictors (Li et al., 2015; Yin et al., 2012). All conclusions have been validated successfully in other cities as New York and Washington. Different scores have been achieved in the three cities. This difference is due to the difference of networks. Most scores are dependent on the traffic volume. Figure 4.24 shows the distribution of station size in the three networks. This figure shows that New York has a lot of small and big stations compared to Washington and Montreal. Washington has smaller stations than Bixi. This difference explains, part of the differences of scores : little stations have less error because of their sizes.

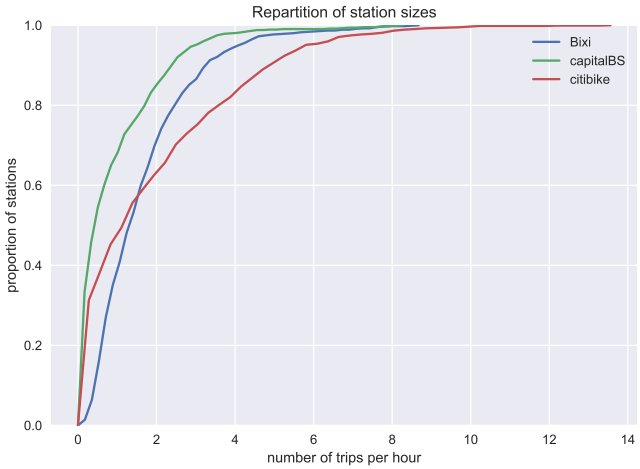

Figure 4.24 Station sizes per network

CHAPTER 5 GENERATING DECISION INTERVALS FOR REBALANCING

Once the bike-sharing (rental and returns) demands are modeled, the next step is to help the operator in its rebalancing decisions. Bixi Montreal currently used decision intervals. These intervals delimit the acceptable fill levels for each station. The operator compares in real time the current inventory to these intervals. If it exits the interval, he dispatches a truck to add or remove bikes from the stations. These intervals provide a rebalancing strategy which is easy to explain and deploy and capable of capturing complex activity patterns. These intervals make it possible to evaluate the distance to the objective differently for stations with low traffic and for stations with high traffic.

Currently, the decision intervals are computed manually using observed demand patterns. Bixi works with five periods per day, one from midnight to 6, one from 6 to 11, one from 11 to 15, one from 15 to 20 and one from 20 to midnight. It defines one interval per station and time-period. Most literature works try to address the rebalancing problem, from an optimization and integer programming point of view (Brinkmann et al., 2015; Chemla et al., 2013; Leduc, 2013; Raviv and Kolka, 2013; Schuijbroek et al., 2013). These papers propose solutions able to solve small to medium instances (up to one hundred stations) with some big approximations (static network, infinite number of trucks, 1 truck, ...). However, there are no studies that solve instances with 500 or more stations. Therefore heuristics need to be used. This chapter proposes an automated way to generate decision intervals, and refines the greedy strategy used by Bixi. Decision intervals (Section 5.2) are generated using the service level (Section 5.1). They are tested using a worst-case analysis (Section 5.2.1) and compared to Bixi ones regarding their performance.

In this chapter we rebuild stations and merge departures and arrivals demand, transforming it in net demand. We compute for each station the distribution probability of the net demand of bikes.

5.1 Service Level

Our approach is inspired by the work of Schuijbroek et al. (2013). In that work the authors define the *service level* as the expected satisfied demand over the expected total demand. The rental and return service levels (Sl_{rent} and Sl_{return}, resp.) of station s with f bikes at

time t_0 for a maximum capacity of C_s are given by :

$$Sl_{rent}(f,s,t_0) = \frac{E(satisfied\ rentals)}{E(rentals)} = \frac{\int_{t_0}^{t_0+T} \mu_s(t)(1-p_s(f,0,t))dt}{\int_{t_0}^{t_0+T} \mu_s(t)dt} \tag{5.1}$$

$$Sl_{return}(f,s,t_0) = \frac{E(satisfied\ returns)}{E(returns)} = \frac{\int_{t_0}^{t_0+T} \lambda_s(t)(1-p_s(f,C_s,t))dt}{\int_{t_0}^{t_0+T} \lambda_s(t)dt}, \tag{5.2}$$

Where $p_s(f,N,t)$ is the probability that station s has N docked bicycles at time t knowing that there were f bikes at time t_0, and $\mu_s(t)$ (resp. $\lambda_s(t)$) is the expected demand at time t and station s for renting (resp. returning) bicycles. Nair et al. (2013) use a similar score in their paper to characterize the quality of a network.

This definition of the service level is very natural as it computes the proportion of satisfied trips. The main criticism (Schuijbroek et al., 2013) made to this definition of the service level is that it does not capture geographical dependence, that influence the service level felt by the customer. This may be problematic as customers tend to use neighboring stations, when a station is in a critical status. The operator will be in charge to handle this dependence.

5.1.1 Modification of the service level

The service level proposed by Schuijbroek evaluates rentals and returns equivalently. However some operators may prioritize satisfied rentals over return or vice versa. For example, an operator that wants to maximize the number of trips in the network may prioritize rentals, and an operator that prioritize more customer satisfaction, may value more returns. Thus, we propose the following modification to the service level definition.

$$Sl_{\alpha}(f,s,t_0) = min(\alpha Sl_{rent}(f,s,t_0), (1-\alpha)Sl_{return}(f,s,t_0)) \tag{5.3}$$

where

$$\alpha = \frac{\text{value of returns}}{\text{value of returns} + \text{value of rentals}} \tag{5.4}$$

is a coefficient of how the operator weights returns versus rentals. An $\alpha = 0.5$ corresponds to definition as used in Schuijbroek. Definition 5.3 gives more flexibility to the operator. The service level is still between 0 and 1. The maximum of $Sl_{\alpha}(f,s,t_0)$ may not be 1, as $\alpha < 1$ and $(1-\alpha) < 1$. This problem could be solved by dividing the service level by $max(\alpha,(1-\alpha))$. However, this normalization is not needed for the application of the next section which considers service levels relatively to the maximum one, in which period and station.

— $\alpha < 0.5$ emphasis is on rentals

— $\alpha > 0.5$ emphasis is on returns

5.1.2 Time Horizon

The service level definition uses a time horizon term T. This term corresponds to the period considered computing the demand estimations. This time horizon influence the value and shape of the service level. A time horizon too small does not consider enough trips and gives a 0/1 service level while a time horizon too big results in a too general service level. Figure 5.1 presents the service levels for several time horizons for the station 6017. Time horizon was taken from 2 hours (gray) to 100 hours (black). For this application, we choose to a time horizon of 10. This time horizon is an upper bound of the prevision time of the operator. After this time horizon, we can consider that the expected future traffic has no influence on the current decision.

5.1.3 Implementation Notes

The complicated part of the service level is the $p_s(f, obj, t)$ probability. This probability needs to be computed for all inventory f at each station, therefore we need to know the probability of each departure and arrival counts, during the time horizon period. This probability is computed from the trip distribution hypothesis. Section 4.7 showed that the best distribution hypothesis was Poisson. Hence the probability $p_s(f, obj, t)$ can be computed as a sum and difference of Poisson processes. This distribution has then a Skellam distribution (Skellam, 1946). It is computed using scipy library. To compute the probability for the other distribution hypothesis we had to compute them manually as no close form formula was available.

5.1.4 Analysis of the Service Level

The service level presents some interesting properties. First, $Sl_{rent}(f, s, t_0)$ and $Sl_{return}(f, s, t_0)$ are monotonous in f, the current capacity of the station. $Sl_{rent}(f, s, t_0)$ is an increasing function of f and $Sl_{return}(f, s, t_0)$ is a decreasing function. This property comes from the probability $p_s(f, 0, t)$ and $p_s(f, C_s, t)$ that are monotonous in f by definition. The Figure 5.2 shows the service level for four stations. Stations 6012 and 6011 correspond to degenerate cases where the return service level is always lower than the rental service level. Their maximum service level is relatively small. Station 6027 is a station with a service level of almost 1 most of the time, and station 6041 is a usual station, with a nice cap $\cap$ shape.

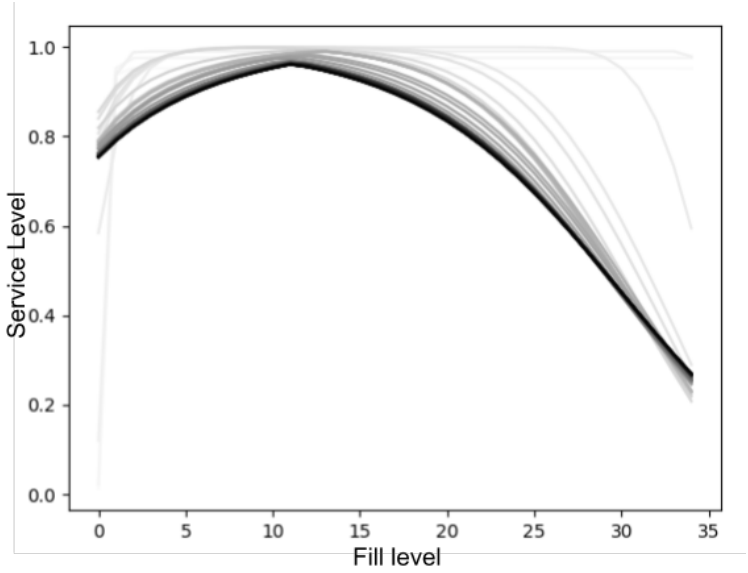

Figure 5.1 Evolution of the service level with different time horizons, from 2 hours (gray) to 100 hours (black)

5.2 Defining decision intervals

Bixi uses intervals for its rebalancing strategy. These intervals change for every period of the week (5 times a day). They are manually redefined every month. They are completed by a target which defines the ideal fill level. This target is not necessarily set as the middle of the interval, it can take any value inside the interval. It is used when rebalancing to define the right number of bikes to add or remove. This section aims to propose an automated process to generate such intervals and targets based on the service level. The service level is defined

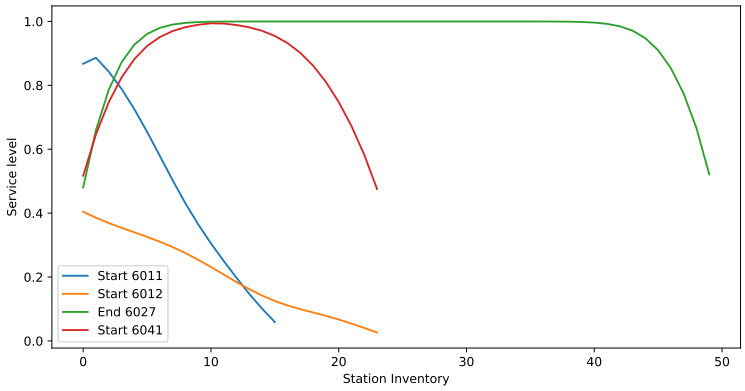

Figure 5.2 Selection of station's service level in the same period

as the proportion of trips we expect to satisfy given a specific fill level. These quantities are related to the urgency of rebalancing a station and therefore to the rebalancing decisions. However, finding the best rebalancing decision is complex, hence a greedy strategy may be adapted. In this section we are interested in generating decision intervals to improve the current decision process of the operator, conserving its current rebalancing strategy. This section gives mathematical background to Bixi decision process.
Decision intervals are composed of a minimum, a maximum and a target. The target value is computed as the maximum service level of the station during the selected period. It maximizes the expected satisfied trips. The following formula computes the minimum and maximum of intervals.

Let's define the minimum and maximum service level of a station s at period t as :

$$SL_{min}(s,t) = min_{x \in \{0..C_s\}}(Sl(x,s,t)) \tag{5.5}$$

$$SL_{max}(s,t) = max_{x \in \{0..C_s\}}(Sl(x,s,t)) \tag{5.6}$$

Then we define $c(s,t)$ the minimum acceptable service level of station s at time period t as :

$$c(s,t) = SL_{min}(s,t) + \beta \cdot (SL_{max}(s,t) - SL_{min}(s,t)) \tag{5.7}$$

Then the interval $I(s,t)$ of station s at period t is computed as

$$I(s,t) = \{x \in \{0..C_s\} | Sl(x,s,t) > c(s,t)\} \tag{5.8}$$

This definition ensures

$$I(s,t) \neq \emptyset \quad \forall s \in S, t \in T$$

Besides, this definition also removes the normalization need for the service level ($Sl < max(\alpha, 1-\alpha)$). The parameter $\beta \in [0,1]$ indicates the requirement level the operator is about the network. A $\beta = 1$ means that stations are requested to stay at their best service level value. A $\beta = 0$ means that stations are free to unbalance. This definition of $c(s,t)$ ensures that intervals are not empty or full ($I(s,t) = \emptyset$ or $I(s,t) = \{0..C_s\}$) unless desired ($\beta = 0$ or $\beta = 1$).

This definition of $c(s,p)$ was built to overcome several limitations of simpler definitions. Simpler definition for this threshold is to ensure a minimum service level for all stations, however, station 46 of Figure 5.3 shows that this definition fails to define an interval for the station : the requested service level cannot be satisfied. Then we chose to satisfy only β

times of the maximum service level of the station : $c(s,t) = SL_{max}(s,t) * \beta$. This definition always defines an interval, however, sometimes the whole interval is selected defining very large intervals for some stations, leading to the absence of rebalancing. Then we defined the threshold to $c(s,t) = min(s,t) + \beta \cdot (max(s,t) - min(s,t))$. The second graph of Figure 5.3 shows that for usual stations, these scores are almost equivalent, then they differ only on degenerated stations (much more departures than arrivals or vice versa).

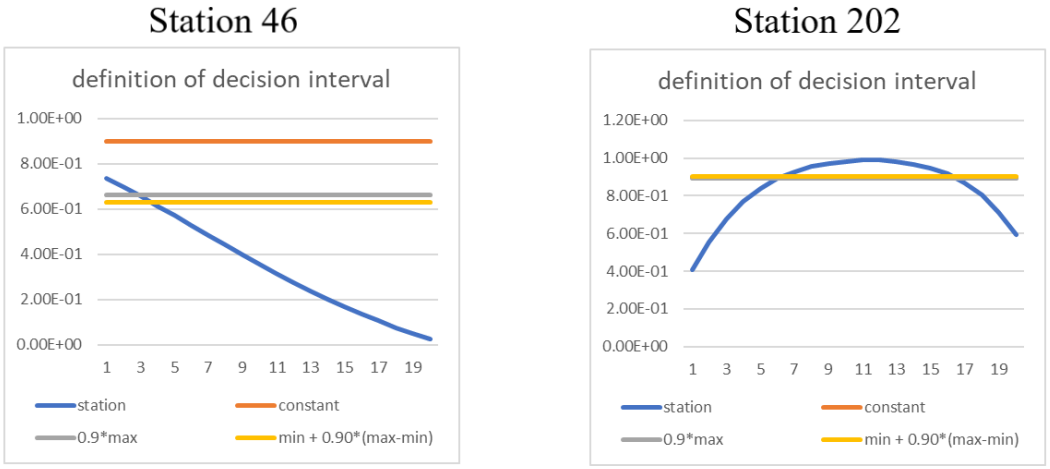

Figure 5.3 Rules to compute decision intervals on station 46 and 202

The β value is chosen manually by the operator. This variable is closely linked to the number of unbalancing alerts raised but the system. Then this value allows the operator to control the constraint imposed to the network. The following sections present a method to define the best β to improve the network service level. This method proposes intervals that meet the rebalancing capacity of the operator, and therefore equivalent to the one used today in terms of the number of rebalancing needed, per day.

5.2.1 Test of decision intervals

Feasibility of intervals

A first test is to check if intervals are feasible for the operator i.e. that the operator can dispatch all its bikes, without violation any interval. First we compute the average of the maximum, minimum and target total number of bikes per hour, respectively. Then we check if the total number of bikes is between the maximum and minimum. These results are shown in table 5.1. Bixi uses 6000 bikes, therefore the operator can dispatch all its bikes. We also note that the sum of targets is slightly lower than the number of bikes. This number seems

reasonable as it counts bicycle in transit. Therefore the proposed best dispatch strategy, counts that around one thousand bicycles are in transit.

Table 5.1 average sum of minimums, targets and maximums per hour

	sum of min	sum of targets	sum of max
Bixi	1483	4833	7825
This work $\beta = 0.9, \alpha = 0.5$	2538	5168	7871

Worst-case analysis

To test the decision intervals, a worst-case analysis is performed. For a given period of time, a simulation is run for each station, we count the number of trips lost if the stations were in the worst fill level possible, inside the interval. More precisely two numbers are computed per station and time period : the number of lost departures and the number of lost arrivals. These two numbers may be positive in the same time period as arrivals and departures are considered separately. Let us define :

$$I_{max}(s,t) = max(i \in I(s,t)) \tag{5.9}$$

$$I_{min}(s,t) = min(i \in I(s,t)) \tag{5.10}$$

The number of departures (resp. arrivals) during period t and station s is noted $d(s,t)$ (resp. $a(s,t)$). Then the average number of lost departures and arrivals are computed as :

$$lost\_dep = mean_{s \in S, t \in T} max(0, (d(s,t) - I_{min}(s,t))) \tag{5.11}$$

$$lost\_arr = mean_{s \in S, t \in T} max(0, (a(s,t) - (C_s - I_{max}(s,t)))) \tag{5.12}$$

For example, for a station of capacity 10, with a decision interval of $[5, 8]$ and 7 departures and 1 arrivals. The lost departures is $max(0, 7-5) = 2$ and the lost arrivals is $max(0, 1-(10-8)) = max(0, -1) = 0$.

Average interval size

To compare decision intervals, this measure (lost trips) is not sufficient as it encourages intervals to be as small as possible : smaller intervals will have better scores. Hence, to compare two set of intervals, we also compare the average length of intervals. Thus, we assume that two set of intervals with similar average sizes are comparable in terms of network constraints.

This assumption suppose that the average size of intervals is highly correlated to the number of alerts, which is not a provable statement. Subsection 5.2.1 proposes intervals with average interval size close to Bixi ones, to propose comparable intervals. Then we compare their performance in terms of worst-case analysis (Subsection 5.2.1).

Target Test

Decision intervals have also a target parameter. To test the goodness of targets, the same worst-case analysis is performed with intervals of size 0. This analysis tests the number of lost trips if the station was at their target fill level at the beginning of each time period.

Rebalancing Constraint

This test tries to quantify the requirement level of the operator on the network. It computes the average number of alerts per station using historical trips. To compute this score, the system is set at time 0 at its target values, then we let it run. Each time a station exits its decision interval, a rebalancing operation is counted, and the station is reset to its target value. Then we count the average number of rebalancing operations needed to keep the network balanced. This test computes the number of times a station must be filled with bikes and the number of times it must be emptied. Algorithm 1 present the pseudo-code of this test. Fist it sets all stations inventory to their target value (line 2), then it computes the net traffic for the whole test period (line 3) and initialize the number of alerts per station at zero ($alert^+$: too many bikes, $alert^-$ not enough bikes, line 4,5). Then for each hour it computes the new inventory of each station (line 8), and test for each station if the inventory is inside the interval (lines 10 and 13) and add one to the number of alerts if the station exit the inventory (lines 11 and 14) and rebalance it (lines 12 and 15). Then the algorithm returns the average number of rebalancing (filling and emptying separately) needed per day during the test period.

Intuitively this test counts the number of rebalancing operations supposing an unlimited truck capacity, that rebalancing operations are immediate, and that all the demand is served.

This algorithm evaluates the strength of the decision interval constraint on the network. The more alerts, the more the network has to be rebalanced to meet the objectives. It is also a measure of the rebalancing capacity needed to satisfy the service level objective a network.

5.2.2 Results on the decision intervals

All the results of this section were computed on the testing data.

Algorithm 1 Mean alert number

```
1: procedure Mean_Alert(Min, Max, Target, departures, arrivals)
2:     inv ← Target[t_0]
3:     net_traffic ← arrivals − departures
4:     alert^+ = Array(size = n_stations, fill = 0)
5:     alert^− = Array(size = n_stations, fill = 0)
6:     days = |hours|/24
7:     for t ∈ hours do
8:         inv = inv + net_traffic[t]
9:         for i ∈ {0, .., size(inv) − 1} do
10:            if inv[i] > Max[t, i] then
11:                alert^+[i] = alert^+[i] + 1
12:                inv[i] = Target[t, i]
13:            if inv[i] < Min[t, i] then
14:                alert^−[i] = alert^−[i] + 1
15:                inv[i] = Target[t, i]
       return sum(alert^+)/days, sum(alert^−)/days
```

Worst case analysis To compare two worst case analyses, we need to compare also the average interval size and the average number of alerts. The target performance is also computed to evaluate its value. Table 5.2 shows these scores on five different sets of intervals : the first one is the set of intervals used by Bixi, the second one is the set of intervals generated using a (SVD,GBT,Poisson) model and a β of 0.5. The third one is a set of intervals generated using the ensemble model (c_3) and a Poisson distribution hypothesis and a β of 0.5. The two-last set of intervals are generated using the two same models but a β of 0.65. The second and third models perform slightly better than Bixi intervals but require one hundred rebalancing per day less (-25%). The fourth and fifth models score better than Bixi intervals, with a comparable number of rebalancing per day. However the target value computed by Bixi seems to be better than the one computed with our algorithm, this value does not depend on β. The Beta parameter had the expected effect : the larger Beta, the thinner the intervals and the better the worst-case score.

β analysis We analyzed the dependency between β and scores. These dependencies are shown in Figure 5.4. This figure shows three graphs, the first one shows the worst-case analysis score, the number of trips lost per hour for different values of β. As expected, the number of lost trips per hour decreases with β. The second graph shows the number of rebalancements per hour needed to keep the network balanced (inside the intervals). This score increases with β as expected : the thinner the intervals, the harder is it to keep the network balanced. The

Table 5.2 Decision interval scores

model	Bixi	SVD GBT P	c_3 P	SVD GBT P	c_3 P
β	-	0.5	0.5	0.65	0.65
lost trips departure	2.80	2.11	2.05	1.66	1.61
lost trips arrivals	2.56	2.93	2.92	2.39	2.38
lost trips	2.68	2.51	2.48	2.02	1.99
lost trips target	0.55	0.68	0.68	0.68	0.68
mean interval size	12.8	16.71	16.57	14.98	14.82
mean alert number	556	417	430	537	552

last graph shows the average size of intervals. The dotted lines represent the value achieved today by Bixi. To generate better intervals than Bixi, we need to choose a β such that the corresponding scores are under Bixi lines.

We choose β such that the generated intervals are equivalent in terms of rebalancing operations. Let's note I_{alerts} the value of β such that $mean\_alerts(\beta) = mean\_alerts\_Bixi$, I_{lost_dep} the value of β such that $lost\_dep(\beta) = lost\_dep\_Bixi$ and I_{lost_arr} the value of β such that $lost\_arr(\beta) = lost\_arr\_Bixi$. Therefore our model will outperform Bixi intervals if $max(I_{lost\_dep}, I_{lost\_arr}) < I_{alerts}$. From Figure 5.4 we note that $I_{alerts} = 0.66$, $I_{lost\_dep} = 0.314$ and $I_{lost\_arr} = 0.60$. We conclude that our intervals outperform Bixi ones. Our intervals allow Bixi to better satisfy its demand, while conserving its rebalancing capacity. Our intervals were also bigger on average than Bixi ones, therefore it is probable that our rebalancing operations concentrate on active stations more than on the inactive ones.

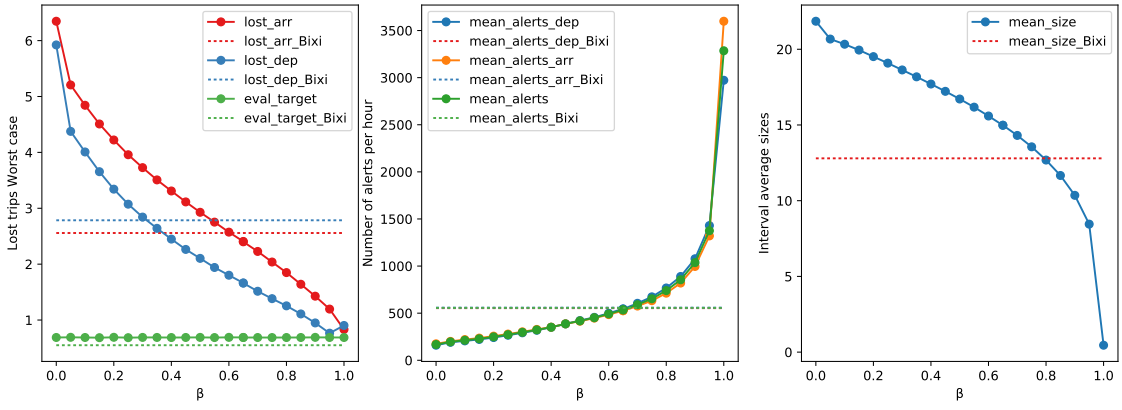

Figure 5.4 Evolution of scores in respect to β ($\alpha = 0.5$)

α analysis All previous tests were made with $\alpha = 0.5$. Then arrivals were considered as important as departures. The operator may want to change this hypothesis to satisfy more

departures than arrivals to maximize the number of trips, or on the opposite to minimize the frustration of users (higher if they can't return a bike than if they can't rent one). Figure 5.5 presents the evolution of scores in respect to α. The dotted lines represent current Bixi performance. The first graph represent the worst-case analysis scores, the departure and arrival graphs intersect at $\alpha = 0.45$. These scores show that the best model is achieved an α of 0.5. If we move away from $\alpha = 0.5$ we trade some arrivals with some departures, however, we lose more trips than we manage to gain. The second graph shows the evolution of the number of rentals, returns and average rebalancing operation needed to keep the network inside the intervals. This score is minimum for a α of 0.5. The β value was chosen to correspond to Bixi's rebalancing capacity (i.e. $\beta = 0.65$), with $\alpha = 0.5$. This graph shows that the value of α does not influence significantly the rebalancing capacity. Then the operator can modify it, without changing its value of β to meet its rebalancing capacity. The last graph shows the evolution of the average interval size. We can clearly see that our intervals are bigger than Bixi ones. Bigger intervals also mean fewer rebalancing operations.

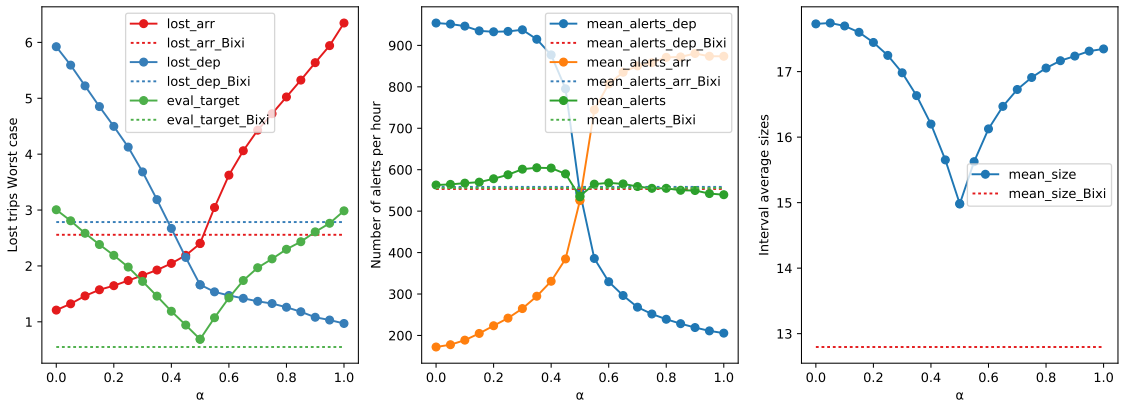

Figure 5.5 Evolution of scores in respect to α ($\beta = 0.65$)

Table 5.3 compares the interval scores between a α of 0.5 and an α of 0.45. This table shows that $\alpha = 0.45$ builds intervals that lose as many rentals as returns. However this trade is costly in terms of the average number of trips lost. This behavior is visible on the first graph of Figure 5.5. Losing one trip less in terms of rentals, means to lose 4 trips in terms of returns, and vice versa.

5.2.3 Estimating the marginal improvement of increasing the rebalancing capacity

The number of alerts determine the rebalancing capacity needed to meet the demand, ensuring a minimum service level. This rebalancing capacity can be linked to the number of

Table 5.3 Influence of α value

model	Bixi	SVD GBT P	c_3 P	SVD GBT P	c_3 P
β	-	0.65		0.65	
α	-	0.5		0.45	
lost trips departure	2.80	1.66	1,61	2.15	2.10
lost trips arrivals	2.55	2.40	2,38	2.19	2.19
lost trips	2.68	2.03	1,99	2.17	2.15
lost trips target	0.55	0.68	0,68	0.94	0.90
mean interval size	12.8	14.97	14,82	15.7	15.49
mean alerts departure	553	546	561	795	801
mean alerts arrivals	558	526	544	384	398
mean alerts	556	536	552	589	599

trips lost, using the worst-case analysis. Figure 5.6 shows the evolution of lost trips in the worst-case analysis as a function of the rebalancing capacity. This figure shows the marginal improvement of the service level, when improving the rebalancing capacity. Bixi current performance is also shown (x marks). This figure can help determine the marginal utility of adding new trucks. This graph first shows that Bixi strategy can already be improved by vertically projecting Bixi points on the graphs. It also helps the operator to quantify the service level improvement of adding a new truck to rebalance the network.

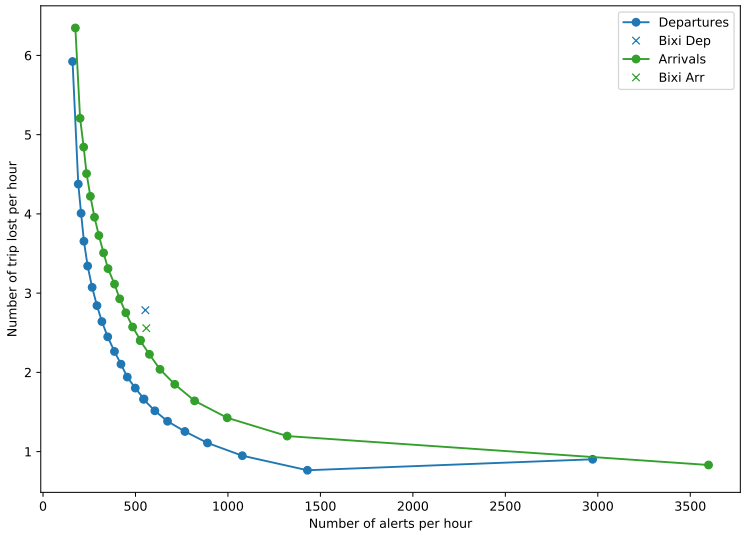

Figure 5.6 Marginal utility of rebalancing

Figure 5.7 shows the number of rebalancing needed each hour of the day, during a week. This figure does not represent the reality exactly as the rebalancing operation is supposed to be instantaneous. However it gives a good insight on when alerts are raised during the day.

Vertical dotted lines separate time-periods, i.e. intervals change at each vertical line. This figure shows that most rebalancing operations happen when intervals are changed and during peak hours. This figure shows that intervals are sufficient inside time periods, except for the peak hours. We can also see that Bixi rebalances too many stations at 3 p.m. and at 8 p.m..

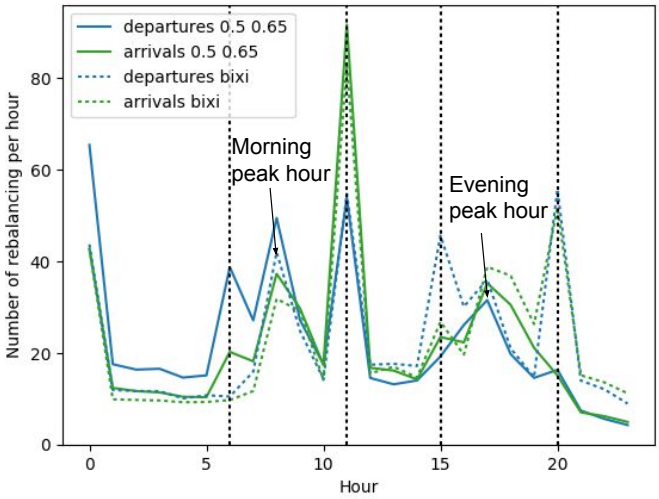

Figure 5.7 Distribution of rebalancing operation during the day

CHAPTER 6 GENERAL DISCUSSION

This part summarizes the work and contributions of the methods explored in each section. It also presents their limitations, proposing some improvements to our model.

Data analysis

The first part of this thesis was devoted to cleaning and analyzing predictive data. We found an issue in the weather data, where half of which was missing. However, the missing data was evenly distributed. In this part we did modeling choices. For example, we have chosen to model traffic by hour. However a shorter time step could be useful. We have also chosen to keep most variables to maximize our ability to predict. However this choice may have impacted the accuracy of some of the tested algorithms.

We also chose to neglect the effect of the station environment, since it can be considered static, and is therefore implicitly learned in the stations' behavior. However this environment is likely to change during the year. We propose next chapter a solution to adapt the network continuously.

We realized in retrospect that some predictive variables have been forgotten, such as Manuvie Free Sundays, which boost traffic once a month. A final criticism that can be made to this part on the choice of the test and training sets. We chose to separate these sets so that they represent continuous and successive time intervals. This choice was made to avoid learning future effects.

Modeling demand per station

The second chapter was devoted to setting up a traffic prediction model. We chose to model only the number of arrivals and departures at each station, and not the complete origin-destination matrix. We felt that it would have been too complex.

The evaluation of the model is also complicated. Indeed every paper of the literature uses its own score. We tried to reconcile all these results, by comparing the performance of our algorithms to the most common scores. We also recalled the relevance of the likelihood to evaluate statistical models.

The peculiarity of our approach has been to try to model the traffic by station, while seeking, first of all, to simplify the problem. We also took a particular interest in determining the pro-

bability distribution of the number of rented/returned bicycles. The literature chose whether to simplify the problem (Chen et al., 2016; Vogel et al., 2011), either to predict the traffic (the number of arrival and departure) by station (Rudloff and Lackner, 2014; Gast et al., 2015) but not both at the same time. Our approach was successful as we managed to effectively reduce the problem (99 %) while gaining in precision. Our model can be summarized by the following formula :

$$P(X_s = k|T = t, W = w) \simeq d(\hat{\mu}_s(t, w), \hat{\sigma}_s(t, w))(k)$$

Where we want to determine as precisely as possible the probability that k bikes are rented during the hour t. This probability is computed by estimating as precisely as possible the distribution d, the expectation $\hat{\mu}_s(t, w)$ and the variance $\hat{\sigma}_s(t, w)$.

Expected demand prediction

The central part of the traffic model is the expected demand prediction. However, this problem is complex because of the number of variables to predict and the stochasticity of the traffic. Thus, to counter these difficulties we propose a two-step approach that first reduces the problem size of 99%, and then learns a predictor. We showed that our approach performs better than the naive approach of building one model per station (Rudloff and Lackner, 2014; Gast et al., 2015), while greatly reducing the computation time. To build our model we compared several approaches. We tried to reduce the problem using some clustering algorithms as kmeans or Gaussian mixture, a SVD reduction and an autoencoder. The SVD reduction obtained the best results in terms of prediction precision and computation time. The autoencoder was able to conserve more information than the SVD but its non-linearity made the extracted behaviors harder to predict, and the inversion of the reduction less robust.

We also explored several prediction methods to model the extracted behaviors. The best algorithms were the gradient boosted tree and the random forest. The neural network was hard to optimize, and we were not able to get better results than with the random forest algorithm. The linear regression was not able to capture some dependencies, even with the introduction of categorical variables. However, we did not test to introduce non-linear transformation of the features.

We completed the model by exploring an ensemble model that learns for each station which is the best algorithms and use that information to improve the results. This algorithm was able to slightly outperform the best algorithm found but required more computation time.

We also proposed two-time series approaches to improve the model, by considering the traffic

and error of the model in the previous time steps. We showed that these models are able to increase slightly the performance of the model. However, the time series approach of the problem was not deeply explored, and some other approaches are still possible, using for example ARIMA and LSTM (neural network) models.

We showed that to predict the demand expectation it was better to use a SVD reduction, with ten dimensions, and a gradient boosted tree predictor. We also showed that more complex models were able to improve slightly the precision but in expense of more computation time. We analyzed our prediction on some stations, and showed that bigger stations are well modeled even if they make bigger errors. Our analysis also suggests that the model could be improved on some stations (small), as the reduction tends to spread the demand by averaging it. To optimize our model, we neglected the dependence between the best hyperparameters and the reduction methods. This approximation introduced some bias in our model and may have influenced our conclusions.

We also tested our model in the New York and Washington networks. These tests confirmed our conclusions. They validated that the SVD-GBT and the SVD-random forest models are the model best suited for traffic prediction and that the ensemble model was able to improve the model precision, but the difference was not always significant.

Expected variance prediction

To complete the demand prediction model, we predicted the expected variance of the demand. We proved that the variance of the demand depends on the features and that some information could be learned. We proposed several models to predict it and concluded that a linear predictor was sufficient as other predictors were too complex and had some issues with overfitting. We were able to explain 4% of the variance. This prediction task had many challenges because of the stochasticity of the squared error and the large number of expected demand predictors. We did not build more complex models because of these challenges, and a further analysis may give better results. We also limit ourselves to the prediction of the variance to determine the parameters of the trips distribution, but we could have predicted other quantities as the probability of zero trips.

Fitting the trip distribution

To complete our traffic model, we fitted a distribution to determine the probability of each trip count. This probability was deduced from the modeled expectation and variance by using a trip distribution hypothesis. This method is very simple to use and accurate. However, it

suffers from imprecision. These distributions are unable to predict or model some rare events that change completely users behaviors. We also limited ourselves to three distributions : the Poisson distribution, the Negative Binomial distribution and the Zero Inflated distribution. They also rely on some hypothesis that can be violated in practice. We did not explore an approach that predicts directly the probability of each trip count for each station. Such an approach could have better results in terms of likelihood.

Decision Intervals

In this dissertation we proposed an application of the bike-sharing traffic model to the rebalancing of the network. We proposed an automated way of computing decision intervals. These intervals define an online rebalancing strategy for the operator. We were able to define flexible and weather dependent intervals that can be adapted to the operator needs. These automatically generated intervals were able to outperform those currently used by Bixi, while preserving the same number of rebalancing operations per day. We analyzed these intervals and point out some weakness in Bixi rebalancing operations. We built a tool to analyze the impact of increasing the rebalancing capacity on the network service level.

CHAPTER 7 CONCLUSION

In this thesis we built a traffic model that predicts the expected distribution of the number of trips per station. The model was conceived to minimize the problem complexity while maximizing the precision of the model. We proved that the original one thousand-dimensional problem could be reduced to a ten-dimensional problem without losing valuable information. This reduction was used to build a predictive model, to compute the expected number of trips per station. Then, this prediction was completed by a distribution hypothesis to compute the probability of the number of trips per hour and per station. We successfully applied this model to automatically compute a weather-dependent online rebalancing strategy, improving Bixi's service level, while maintaining an equivalent rebalancing capacity. We proposed some methods to achieve better performance. However, these methods require more computational power. Furthermore, our model performance was better than other works interested in the traffic prediction per station.

successfully outperformed the state of the art in terms of model precision and model complexity.

Future works

This dissertation focused on building a traffic model for bike-sharing systems and providing a direct application of the model. However, several research problems and ideas were raised and not fully exploited. The most notable ones are to modify the model to adapt it to network changes, and to use it to automate the rebalancing operations.

Predict unmet demand

The model built in this work did not consider stations in a critical status (full or empty), and then did not estimate the unmet demand. This unmet demand may have biased the model as it was not registered in the trip information. This effect is probably limited as the reduction part erased part of this bias. However, it is important to estimate it to get a correct traffic model and to improve the performance of the actual one.

Adding new stations and correcting the model

Adding stations to a bike-sharing traffic model or updating the model so that it considers short term changes can be challenging as few data is available. Our model offers an easy

and elegant way of dealing with this problem, using a two-step approach. First it learns the models built in this work, then it uses recent data to rebuild the *inv_red* method. This new learning phase uses the prediction of the first algorithm as features, and the real traffic as objectives. This method transforms the addition of a station into finding the best linear

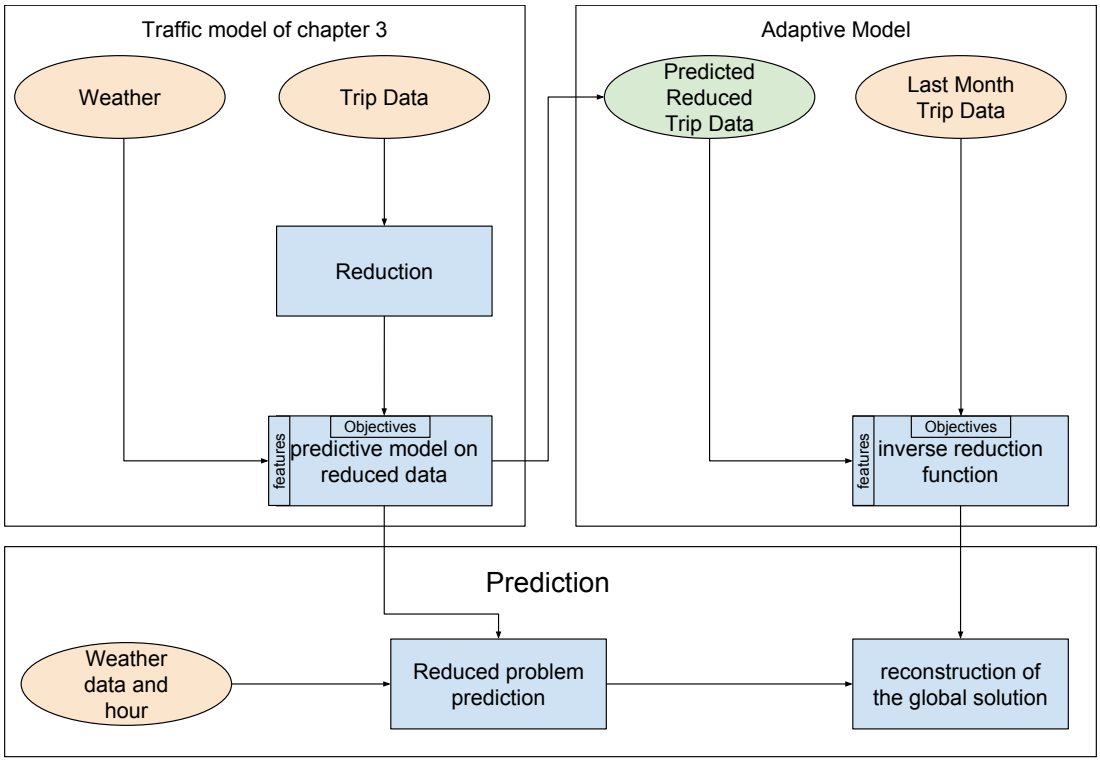

Figure 7.1 Adaptive model flowsheet, on the top left box the learning of the predictive algorithm, on the top right box the learning of the adaptive part and the bottom the global prediction process

combination of the extracted behaviors, that explains the station traffic. These parameters could be estimated using about three weeks of data. Therefore, with only three weeks of data we are able to build a complete model for a new station that takes into account weather and seasonality effects. This approach also builds a traffic model able to adapt itself to short term modifications of the network and user behavior. This modification of the algorithm can be very useful in Montreal where stations are moved quite often because of the road works.

Automating the rebalancing operation

The main application of the traffic model built in this work is to automate the rebalancing operations. In this work we proposed an online rebalancing strategy. However, this strategy

is not complete, if too many alerts are raised, the operator has to choose which one to solve and which one to ignore. A heuristic to solve this problem is to rank these alerts and always to solve the top priority alert first. The rank can be computed combining the service level gain for rebalancing the station, the number of bikes missing, the distance of the nearest truck, etc...

Another approach to automate the rebalancing decision is to learn a reinforcement learning algorithm. This algorithm will learn the best decision to take, using the model build in this work to estimate the state transition probability matrix and the rewards.

REFERENCES

"Donnée meteorologiques horaires", http://climat.meteo.gc.ca/historical_data/search_historic_data_f.html, 2018.

2014. En ligne : https://www.kaggle.com/c/bike-sharing-demand

D. Aloise, A. Deshpande, P. Hansen, et P. Popat, "Np-hardness of euclidean sum-of-squares clustering", *Machine learning*, vol. 75, no. 2, pp. 245–248, 2009.

R. Alvarez-Valdes, J. M. Belenguer, E. Benavent, J. D. Bermudez, F. Muñoz, E. Vercher, et F. Verdejo, "Optimizing the level of service quality of a bike-sharing system", *Omega*, vol. 62, pp. 163–175, 2016.

S. Anily et R. Hassin, "The swapping problem", *Networks*, vol. 22, no. 4, pp. 419–433, 1992.

P. Baldi et K. Hornik, "Neural networks and principal component analysis : Learning from examples without local minima", *Neural Networks*, vol. 2, no. 1, pp. 53 – 58, 1989. DOI : https://doi.org/10.1016/0893-6080(89)90014-2. En ligne : http://www.sciencedirect.com/science/article/pii/0893608089900142

M. Benchimol, P. Benchimol, B. Chappert, A. De La Taille, F. Laroche, F. Meunier, et L. Robinet, "Balancing the stations of a self service "bike hire" system", *RAIRO-Operations Research*, vol. 45, no. 1, pp. 37–61, 2011.

H. Bensusan et C. Giraud-Carrier, "Discovering task neighbourhoods through landmark learning performances", dans *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2000, pp. 325–330.

G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, et G. Laporte, "Static pickup and delivery problems : aclassification scheme and survey", *TOP*, vol. 15, no. 1, pp. 1–31, 2007. DOI : 10.1007/s11750-007-0009-0. En ligne : http://dx.doi.org/10.1007/s11750-007-0009-0

M. Bordagaray, L. dell'Olio, A. Fonzone, et Á. Ibeas, "Capturing the conditions that introduce systematic variation in bike-sharing travel behavior using data mining techniques", *Transportation research part C : emerging technologies*, vol. 71, pp. 231–248, 2016.

P. Borgnat, P. Abry, P. Flandrin, C. Robardet, J.-B. Rouquier, et E. Fleury, "Shared bicycles

in a city : A signal processing and data analysis perspective", *Advances in Complex Systems*, vol. 14, no. 03, pp. 415–438, 2011.

L. Breiman, "Bagging predictors", *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

J. Brinkmann, M. W. Ulmer, et D. C. Mattfeld, "Short-term strategies for stochastic inventory routing in bike sharing systems", *Transportation Research Procedia*, vol. 10, pp. 364–373, 2015.

L. Caggiani et M. Ottomanelli, "A modular soft computing based method for vehicles repositioning in bike-sharing systems", *Procedia-Social and Behavioral Sciences*, vol. 54, pp. 675–684, 2012.

L. Cagliero, T. Cerquitelli, S. Chiusano, P. Garza, et X. Xiao, "Predicting critical conditions in bicycle sharing systems", *Computing*, vol. 99, no. 1, pp. 39–57, 2017.

D. Chemla, "Algorithms for optimized shared transport systems", Thesis, 2012.

D. Chemla, F. Meunier, et R. W. Calvo, "Bike sharing systems : Solving the static rebalancing problem", *Discrete Optimization*, vol. 10, no. 2, pp. 120–146, 2013.

L. Chen, D. Zhang, L. Wang, D. Yang, X. Ma, S. Li, Z. Wu, G. Pan, T.-M.-T. Nguyen, et J. Jakubowicz, "Dynamic cluster-based over-demand prediction in bike sharing systems", dans *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2016, pp. 841–852.

C. Contardo, C. Morency, et L.-M. Rousseau, *Balancing a dynamic public bike-sharing system*. Cirrelt Montreal, 2012, vol. 4.

T. Dozat, "Incorporating nesterov momentum into adam", 2016.

C. Etienne et O. Latifa, "Model-based count series clustering for bike sharing system usage mining : a case study with the vélib'system of paris", *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 3, p. 39, 2014.

A. Faghih-Imani et N. Eluru, "Incorporating the impact of spatio-temporal interactions on bicycle sharing system demand : A case study of new york citibike system", *Journal of Transport Geography*, vol. 54, pp. 218–227, 2016.

H. Fanaee-T et J. Gama, "Event labeling combining ensemble detectors and background knowledge", *Progress in Artificial Intelligence*, vol. 2, no. 2-3, pp. 113–127, 2014.

U. M. Fayyad et K. B. Irani, “On the handling of continuous-valued attributes in decision tree generation”, *Machine learning*, vol. 8, no. 1, pp. 87–102, 1992.

C. Fricker et N. Gast, “Incentives and redistribution in bike-sharing systems with stations of finite capacity”, 2013.

J. C. García-Palomares, J. Gutiérrez, et M. Latorre, “Optimizing the location of stations in bike-sharing programs : a gis approach”, *Applied Geography*, vol. 35, no. 1, pp. 235–246, 2012.

N. Gast, G. Massonnet, D. Reijsbergen, et M. Tribastone, “Probabilistic forecasts of bike-sharing systems for journey planning”, dans *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 2015, pp. 703–712.

K. Gebhart et R. B. Noland, “The impact of weather conditions on bikeshare trips in washington, dc”, *Transportation*, vol. 41, no. 6, pp. 1205–1225, 2014.

S. Ghosh, M. Trick, et P. Varakantham, “Robust repositioning to counter unpredictable demand in bike sharing systems”, dans *International Joint Conference on Artificial Intelligence (IJCAI)*, Conference Proceedings.

R. C. Hampshire et L. Marla, “An analysis of bike sharing usage : Explaining trip generation and attraction from observed demand”, dans *91st Annual meeting of the transportation research board, Washington, DC*, 2012, Conference Proceedings, pp. 12–2099.

K. He, X. Zhang, S. Ren, et J. Sun, “Deep residual learning for image recognition”, dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

H. Hernández-Pérez et J.-J. Salazar-González, “A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery”, *Discrete Applied Mathematics*, vol. 145, no. 1, pp. 126–139, 2004.

G. E. Hinton et R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks”, *science*, vol. 313, no. 5786, pp. 504–507, 2006.

D. P. Kingma et J. Ba, “Adam : A method for stochastic optimization”, *arXiv preprint arXiv :1412.6980*, 2014.

N. Lathia, S. Ahmed, et L. Capra, “Measuring the impact of opening the london shared bicycle scheme to casual users”, *Transportation research part C : emerging technologies*,

vol. 22, pp. 88–102, 2012.

A. Leduc, “Modèle d’optimisation de la redistribution des vélos d’un système de vélopartage”, Thesis, 2013.

I. Leontaritis et S. A. Billings, “Input-output parametric models for non-linear systems part i : deterministic non-linear systems”, *International journal of control*, vol. 41, no. 2, pp. 303–328, 1985.

Y. Li, Y. Zheng, H. Zhang, et L. Chen, “Traffic prediction in a bike-sharing system”, dans *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2015, p. 33.

J.-R. Lin et T.-H. Yang, “Strategic design of public bicycle sharing systems with service level constraints”, *Transportation research part E : logistics and transportation review*, vol. 47, no. 2, pp. 284–294, 2011.

J.-H. Lin et T.-C. Chou, “A geo-aware and vrp-based public bicycle redistribution system”, *International Journal of Vehicular Technology*, vol. 2012, 2012.

M. Lowalekar, P. Varakantham, S. Ghosh, S. D. Jena, et P. Jaillet, “Online repositioning in bike sharing systems”, dans *International Conference on Automated Planning and Scheduling (ICAPS)*, 2017.

M. S. Mahmoud, W. El-Assi, et K. N. Habib, “Effects of built environment and weather on bike sharing demand : Station level analysis of commercial bike sharing in toronto”, dans *94th Transportation Research Board Annual Meeting, Washington, DC*, Conference Proceedings.

L. M. Martinez, L. Caetano, T. Eiró, et F. Cruz, “An optimisation algorithm to establish the location of stations of a mixed fleet biking system : an application to the city of lisbon”, *Procedia-Social and Behavioral Sciences*, vol. 54, pp. 513–524, 2012.

L. D. MEng, “Implementing bike-sharing systems”, *Proceedings of the Institution of Civil Engineers*, vol. 164, no. 2, p. 89, 2011.

R. Nair, E. Miller-Hooks, R. C. Hampshire, et A. Bušić, “Large-scale vehicle sharing systems : analysis of vélib”’, *International Journal of Sustainable Transportation*, vol. 7, no. 1, pp. 85–106, 2013.

N. G. Polson et V. O. Sokolov, “Deep learning for short-term traffic flow prediction”, *Transportation Research Part C : Emerging Technologies*, vol. 79, pp. 1–17, 2017.

T. Raviv et O. Kolka, “Optimal inventory management of a bike-sharing station”, *IIE Transactions*, vol. 45, no. 10, pp. 1077–1093, 2013.

T. Raviv, M. Tzur, et I. A. Forma, “Static repositioning in a bike-sharing system : models and solution approaches”, *EURO Journal on Transportation and Logistics*, vol. 2, no. 3, pp. 187–229, 2013.

C. Rudloff et B. Lackner, “Modeling demand for bikesharing systems : neighboring stations as source for demand and reason for structural breaks”, *Transportation Research Record : Journal of the Transportation Research Board*, no. 2430, pp. 1–11, 2014.

J. Schuijbroek, R. Hampshire, et W.-J. van Hoeve, “Inventory rebalancing and vehicle routing in bike sharing systems”, 2013.

J. Shu, M. Chou, Q. Liu, C.-P. Teo, et I.-L. Wang, “Bicycle-sharing system : deployment, utilization and the value of re-distribution”, *National University of Singapore-NUS Business School, Singapore*, 2010.

J. G. Skellam, “The frequency distribution of the difference between two poisson variates belonging to different populations”, *Journal of the Royal Statistical Society : Series A*, vol. 109, p. 296, 1946.

E. Slutzky, “The summation of random causes as the source of cyclic processes”, *Econometrica : Journal of the Econometric Society*, pp. 105–146, 1937.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, et R. Salakhutdinov, “Dropout : A simple way to prevent neural networks from overfitting”, *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

P. Vogel et D. Mattfeld, “Strategic and operational planning of bike-sharing systems by data mining–a case study”, *Computational Logistics*, pp. 127–141, 2011.

P. Vogel et D. C. Mattfeld, “Modeling of repositioning activities in bike-sharing systems”, dans *World conference on transport research (WCTR)*, 2010.

P. Vogel, T. Greiser, et D. C. Mattfeld, “Understanding bike-sharing systems using data mining : Exploring activity patterns”, *Procedia-Social and Behavioral Sciences*, vol. 20, pp. 514–523, 2011.

P. Vogel, B. A. N. Saavedra, et D. C. Mattfeld, “A hybrid metaheuristic to solve the resource allocation problem in bike sharing systems”, dans *International Workshop on Hybrid Metaheuristics*. Springer, 2014, pp. 16–29.

P. Vogel, J. F. Ehmkeb, et D. C. Mattfelda, *Service network design of bike sharing systems*. Springer, 2016, pp. 113–135.

W. Wang, “Forecasting bike rental demand using new york citi bike data”, 2016.

A. Waserhole et V. Jost, “Vehicle sharing system pricing regulation : Transit optimization of intractable queuing network. hal-00751744”, 2012.

Y.-C. Yin, C.-S. Lee, et Y.-P. Wong, “Demand prediction of bicycle sharing systems”, 2012.

J. W. Yoon, F. Pinelli, et F. Calabrese, “Cityride : a predictive bike sharing journey advisor”, dans *Mobile Data Management (MDM), 2012 IEEE 13th International Conference on*. IEEE, 2012, pp. 306–311.

G. U. Yule *et al.*, “Vii. on a method of investigating periodicities disturbed series, with special reference to wolfer’s sunspot numbers”, *Phil. Trans. R. Soc. Lond. A*, vol. 226, no. 636-646, pp. 267–298, 1927.

J. Zhang, X. Pan, M. Li, et S. Y. Philip, “Bicycle-sharing system analysis and trip prediction”, dans *Mobile Data Management (MDM), 2016 17th IEEE International Conference on*, vol. 1. IEEE, 2016, pp. 174–179.

Y. Zhou, L. Wang, R. Zhong, et Y. Tan, “A markov chain based demand prediction model for stations in bike sharing systems”, *Mathematical Problems in Engineering*, vol. 2018, 2018.

APPENDIX A PRECISION RESULTS

Table A.1 Precision scores for mean estimation algorithms on validation set

Reduction	Prediction	rmsle	rmse	mae	r2	mape	train_time	test_time
svd	gbt	0.487	1.881	1.022	0.662	0.427	15.109	0.147
autoencoder	gbt	0.487	1.902	1.004	0.654	0.402	10.783	1.783
svd	randforest	0.492	1.925	1.039	0.646	0.443	2.407	0.293
autoencoder	randforest	0.489	1.927	1.016	0.645	0.412	7.114	3.409
autoencoder	decisiontree	0.498	1.981	1.042	0.625	0.423	6.321	3.988
svd	decisiontree	0.499	1.99	1.06	0.622	0.446	1.199	0.088
GM	gbt	0.505	2.007	1.087	0.615	0.465	13.658	0.202
kmeans	gbt	0.505	2.006	1.088	0.615	0.464	13.159	0.165
GM	randforest	0.504	2.017	1.087	0.611	0.464	2.404	0.439
kmeans	randforest	0.505	2.018	1.089	0.611	0.467	2.466	0.322
GM	decisiontree	0.508	2.06	1.1	0.595	0.464	1.422	0.1
kmeans	decisiontree	0.509	2.059	1.103	0.595	0.467	1.36	0.092
svd	linear	0.602	2.438	1.326	0.432	0.612	2.099	0.083
svd	MLP	0.607	2.447	1.338	0.428	0.624	65.95	5.297
GM	linear	0.615	2.474	1.363	0.415	0.641	1.947	0.105
kmeans	linear	0.615	2.475	1.364	0.415	0.642	13.453	0.101
kmeans	MLP	0.602	2.48	1.334	0.412	0.603	67.823	7.156
autoencoder	MLP	0.567	2.497	1.23	0.404	0.458	59.364	5.79
autoencoder	linear	0.567	2.518	1.225	0.394	0.444	3.515	1.344
GM	MLP	0.653	2.528	1.451	0.389	0.73	71.975	10.197
GM	mean	0.664	2.859	1.511	0.219	0.641	1.35	0.134
kmeans	mean	0.664	2.859	1.511	0.219	0.641	1.294	0.105
svd	mean	0.664	2.859	1.511	0.219	0.64	1.191	0.09
sum	randforest	0.741	2.986	1.697	0.148	0.839	1.97	0.196
sum	gbt	0.744	2.987	1.706	0.147	0.85	2.665	0.12
sum	decisiontree	0.745	2.997	1.704	0.142	0.842	1.283	0.088
sum	linear	0.799	3.103	1.85	0.08	0.972	1.314	0.094
sum	MLP	0.8	3.104	1.854	0.08	0.976	65.262	6.436
autoencoder	mean	0.708	3.224	1.532	0.007	0.436	5.339	2.976
sum	mean	0.805	3.257	1.869	-0.014	0.874	1.181	0.09

Score of algorithms on the test set

svd-gbt	0.424	1.268	0.65	R^2	-1	0.323	20.84	0.11
c4-c4	0.434	1.28	0.681	0.617	-0.936	0.369	0	13.794
c3-c3	0.435	1.281	0.682	0.616	-0.939	0.371	0	11.979
c2-c2	0.434	1.283	0.682	0.615	-0.944	0.369	0	11.797
c5-c5	0.434	1.285	0.683	0.614	-0.936	0.371	0	11.373
svd-randforest	0.436	1.29	0.695	0.611	-0.942	0.379	2.944	0.265
id-gbt	0.446	1.291	0.699	0.61	-0.992	0.382	1407.743	4.041
id-randforest	0.445	1.328	0.712	0.588	-0.953	0.393	55.209	0.557
c1-c1	0.451	1.352	0.712	0.573	-0.985	0.389	0	15.736
svd-decisiontree	0.444	1.355	0.713	0.571	-0.973	0.38	1.719	0.072
GM-gbt	0.443	1.373	0.699	0.56	-1.018	0.356	19.155	0.132
kmeans-randforest	0.443	1.375	0.721	0.558	-0.966	0.385	3.343	0.27
kmeans-gbt	0.442	1.378	0.696	0.556	-1.031	0.35	19.641	0.106
GM-randforest	0.462	1.405	0.754	0.539	-0.983	0.423	3.188	0.273
kmeans-decisiontree	0.451	1.408	0.733	0.536	-0.99	0.386	1.724	0.062
id-decisiontree	0.461	1.416	0.737	0.532	-1.026	0.396	2.909	0.044
autoencoder-gbt	0.455	1.42	0.698	0.529	-1.106	0.354	19.33	5.242
autoencoder-randforest	0.481	1.464	0.762	0.499	-1.05	0.427	11.729	5.498
GM-decisiontree	0.499	1.506	0.821	0.47	-1.036	0.479	1.739	0.071
autoencoder-decisiontree	0.498	1.554	0.803	0.435	-1.088	0.452	10.578	5.207
svd-linear	0.516	1.662	0.844	0.354	-1.248	0.441	2.174	0.068
svd-ridge	0.516	1.662	0.844	0.354	-1.247	0.442	1.983	0.055
autoencoder-meanhour	0.537	1.667	0.883	0.35	-1.168	0.511	5.427	1.773
id-linear	0.518	1.667	0.845	0.35	-1.26	0.442	2.048	0.049

algorithm	$RMSLE$	$RMSE$	MAE	R^2	Average ll Poisson	$MAPE$	train (sec)	test (sec)
id-ridge	0.518	1.667	0.845	0.35	-1.259	0.442	1.72	0.065
GM-linear	0.519	1.682	0.853	0.339	-1.25	0.446	2.37	0.063
kmeans-linear	0.52	1.684	0.854	0.337	-1.249	0.448	14.431	0.072
kmeans-ridge	0.52	1.684	0.854	0.337	-1.249	0.448	13.615	0.057
GM-ridge	0.52	1.684	0.854	0.337	-1.252	0.447	2.216	0.057
svd-lasso	0.507	1.704	0.815	0.321	-1.331	0.388	1.547	0.049
id-meanhour	0.569	1.745	0.973	0.288	-1.145	0.613	5.742	0.042
svd-meanhour	0.571	1.75	0.978	0.284	-1.151	0.617	1.468	0.057
id-lasso	0.594	1.75	1.021	0.284	-1.297	0.636	89.459	0.054
GM-meanhour	0.576	1.761	0.992	0.275	-1.165	0.626	1.675	0.06
kmeans-meanhour	0.576	1.762	0.992	0.274	-1.165	0.626	1.59	0.055
autoencoder-linear	0.541	1.853	0.822	0.197	-1.801	0.319	9.078	4.414
autoencoder-ridge	0.541	1.882	0.809	0.172	-1.881	0.286	156.929	0.805
kmeans-lasso	0.654	1.89	1.149	0.165	-1.379	0.758	2.225	0.056
GM-lasso	0.654	1.89	1.149	0.165	-1.379	0.758	2.387	0.053
sum-gbt	0.601	1.899	1.021	0.157	-1.443	0.575	3.186	0.061
sum-randforest	0.593	1.899	1.035	0.157	-1.408	0.584	2.52	0.17
sum-decisiontree	0.6	1.908	1.046	0.149	-1.425	0.593	1.656	0.058
autoencoder-lasso	0.539	1.912	0.809	0.146	-1.856	0.275	4.964	1.13
svd-mean	0.687	1.982	1.223	0.082	-1.45	0.822	1.438	0.048
kmeans-mean	0.688	1.982	1.224	0.082	-1.451	0.823	1.545	0.056
GM-mean	0.688	1.982	1.224	0.082	-1.451	0.823	1.628	0.055
id-mean	0.688	1.982	1.224	0.082	-1.451	0.823	1.632	0.044
autoencoder-mean	0.559	1.993	0.917	0.071	-1.656	0.373	5.039	1.538
sum-linear	0.633	2	1.087	0.065	-1.662	0.608	1.428	0.052

algorithm	$RMSLE$	$RMSE$	MAE	R^2	Average ll Poisson	$MAPE$	train (sec)	test (sec)
sum-ridge	0.633	2	1.087	0.065	-1.662	0.608	1.553	0.053
sum-lasso	0.632	2.002	1.078	0.064	-1.687	0.595	1.44	0.064
sum-meanhour	0.729	2.046	1.324	0.021	-1.605	0.906	1.386	0.054
sum-mean	0.785	2.124	1.474	-0.054	-1.843	1.026	1.505	0.06

APPENDIX B RESULTS ON MONTREAL, WASHINGTON AND NEW YORK

Scores on Bixi data

Table B.1 Scores on Montreal (Bixi) without weather features

Algorithm	$RMSLE$	$RMSE$	MAE	R^2	Log Likelihood	$MAPE$	test (sec)	train (sec)
svd, gbt	0.494	1.483	0.804	0.486	-1.035	0.469	0.071	11.107
kmeans, gbt	0.5	1.524	0.817	0.457	-1.059	0.472	0.075	11.338
GM, gbt	0.504	1.527	0.825	0.455	-1.061	0.482	0.077	11.368
autoencoder, gbt	0.499	1.558	0.8	0.432	-1.091	0.449	1.12	13.455
GM, randforest	0.474	1.584	0.795	0.414	-1.025	0.427	0.153	1.489
kmeans, randforest	0.476	1.601	0.801	0.401	-1.027	0.432	0.153	1.521
svd, randforest	0.502	1.621	0.846	0.386	-1.043	0.489	0.148	1.417
kmeans, decisiontree	0.471	1.657	0.803	0.358	-1.05	0.408	0.048	0.788
GM, decisiontree	0.472	1.663	0.805	0.354	-1.053	0.409	0.051	0.771
svd, decisiontree	0.504	1.683	0.859	0.338	-1.069	0.479	0.041	0.734
autoencoder, randforest	0.564	1.883	0.966	0.172	-1.18	0.582	1.579	4.639
autoencoder, decisiontree	0.571	1.913	0.98	0.145	-1.195	0.595	2.099	4.105
autoencoder, linear	0.596	1.992	1.029	0.073	-1.436	0.568	0.879	156.518
svd, linear	0.678	2.018	1.226	0.048	-1.407	0.822	0.042	1.317
GM, linear	0.688	2.034	1.247	0.033	-1.426	0.843	0.051	1.327
kmeans, linear	0.688	2.038	1.249	0.029	-1.427	0.844	0.048	11.913

Scores on Capital Bikeshare data

Table B.2 Scores on Washington (Capital Bikeshare)

Algorithm	$RMSLE$	$RMSE$	MAE	R^2	Log Likelihood	$MAPE$	train_time	test_time
svd, gbt	0.385	1.269	0.531	0.601	-0.823	0.253	12.619	0.236
svd, randforest	0.383	1.277	0.545	0.596	-0.75	0.28	2.408	0.441
svd, decisiontree	0.395	1.342	0.56	0.554	-0.793	0.286	1.363	0.101
autoencoder, decisiontree	0.415	1.413	0.573	0.505	-inf	0.282	4.792	2.107
svd, linear	0.566	1.835	0.898	0.166	-1.074	0.567	2.265	0.07
kmeans, gbt	0.481	1.876	0.745	0.129	-1.052	0.384	12.036	0.148
GM, gbt	0.467	1.882	0.714	0.123	-inf	0.364	12.989	0.174
autoencoder, randforest	0.578	2.000	0.692	0.009	-inf	0.223	5.077	3.156
autoencoder, gbt	0.608	2.105	0.726	-0.097	-inf	0.221	14.432	1.391
GM, linear	0.641	2.118	1.07	-0.112	-1.23	0.715	1.97	0.089
kmeans, linear	0.642	2.126	1.074	-0.119	-1.233	0.717	13.08	0.09
autoencoder, linear	0.603	2.436	0.927	-0.47	-inf	0.47	293.457	1.024
GM, randforest	0.559	2.955	1.023	-1.164	-1.084	0.578	2.282	0.197
GM, decisiontree	0.569	2.993	1.042	-1.219	-1.111	0.595	1.225	0.089
kmeans, randforest	0.577	3.027	1.08	-1.27	-1.125	0.614	2.22	0.198
kmeans, decisiontree	0.582	3.055	1.089	-1.312	-1.138	0.621	1.201	0.098

Scores on CityBike data

Table B.3 Scores on CityBike (New York)

Algorithm	$RMSLE$	$RMSE$	MAE	R^2	Log Likelihood	$MAPE$	train_time	test_time
svd, gbt	0.624	3.106	1.267	0.439	-2.43	0.398	14.441	0.24
svd, randforest	0.586	3.206	1.323	0.402	-1.584	0.532	3.39	0.258
svd, decisiontree	0.604	3.262	1.38	0.381	-1.585	0.612	2.087	0.157
svd, linear	0.848	3.944	2.039	0.095	-2.204	1.146	3.99	0.154
autoencoder, gbt	0.952	4.485	1.686	-0.17	-inf	0.361	16.445	1.522
autoencoder, randforest	0.955	4.501	1.695	-0.178	-inf	0.367	7.245	2.469
autoencoder, decisiontree	0.911	4.516	1.692	-0.186	-inf	0.414	6.57	2.729
GM, linear	1.003	5.087	2.762	-0.505	-2.856	1.644	4.653	0.238
kmeans, linear	1.015	5.307	2.855	-0.638	-2.93	1.7	14.626	0.204
kmeans, randforest	0.778	5.549	2.12	-0.791	-inf	0.856	3.485	0.315
autoencoder, linear	0.822	5.623	2.209	-0.839	-inf	0.919	419.805	1.12
GM, randforest	0.748	5.651	1.885	-0.857	-inf	0.721	3.455	0.308
kmeans, decisiontree	1.004	5.941	2.292	-1.053	-inf	0.752	2.262	0.202
GM, decisiontree	1.007	6.153	2.249	-1.203	-inf	0.739	2.283	0.204
kmeans, gbt	0.821	7.135	2.553	-1.961	-3.263	0.996	14.702	0.266
GM, gbt	0.831	7.9	2.739	-2.631	-3.235	1.075	14.511	0.26